HEALTHCARE AI AGENTS

Amna khan

GHUNCHAS

Plugins & Extensibility

Healthcare Agent Service is designed to be highly extensible. Plugins allow the agent to go beyond static conversations and interact dynamically with data sources, systems, and external services.

Plugin Architecture & Purpose

Plugins are modular components that extend the capabilities of a healthcare agent. They enable the agent to retrieve information, perform actions, or generate responses using external or internal systems.

Purpose of Plugins

Plugins allow healthcare agents to:

- Access private or public knowledge sources
- Integrate with enterprise systems and APIs
- Perform dynamic actions during conversations
- Deliver contextual, real-time healthcare responses

This modular approach makes the system flexible and scalable.

Plugin Registration Process

Before a plugin can be used:

1. It must be registered in the Healthcare Agent Service
2. Required credentials and endpoints are configured
3. Metadata is defined to describe plugin functionality

Once registered, the plugin becomes available to the Healthcare Orchestrator.

Plugin Metadata and Descriptions

Each plugin includes metadata such as:

- Plugin name and description
- Trigger conditions
- Type of data or action provided

This metadata helps the orchestrator decide when and how to use a plugin during a conversation.

Supported Plugin Types

Healthcare Agent Service supports multiple plugin types to cover different healthcare scenarios.

Generative Answers on Customer Sources

These plugins allow the agent to:

- Use private organizational data
- Retrieve information via Azure AI Search
- Generate grounded, secure responses

Common use cases include hospital policies, internal protocols, and patient education material.

Generative Answers on Built-in Sources

Built-in plugins provide access to:

- Platform-managed medical knowledge
- Trusted healthcare intelligence

These plugins ensure baseline medical accuracy even when customer data is unavailable.

Conversational Plugins

Conversational plugins allow:

- Dynamic question handling
- Task execution during conversations
- Multi-step workflows

They are useful for appointment scheduling, patient intake, and support workflows.

OpenAPI Plugins

OpenAPI plugins connect external services using standardized APIs.
They enable deep integration with healthcare systems such as EHRs and scheduling platforms.

OpenAPI (Swagger) Fundamentals

OpenAPI defines a standard way to describe REST APIs, including:

- Endpoints
- Request formats
- Response structures

This ensures consistent and predictable integrations.

External Service Integration

OpenAPI plugins enable secure integration with:

- Electronic Health Record (EHR) systems
- Scheduling systems
- Billing or insurance services

This allows agents to move from information delivery to real-world action.

Structured Input and Output Handling

Inputs and outputs are structured using JSON schemas, ensuring:

- Reliable data exchange
- Validation of responses
- Error handling consistency

Orchestrator Decision Enhancement

By exposing capabilities through OpenAPI plugins, the orchestrator gains more options to respond intelligently based on user intent.

Data Connections & Advanced Functionality

Data connections form the backbone of secure communication between the agent and external systems.

Data Connections Overview

Data connections define how the agent communicates with external services.

Key characteristics include:

- Secure HTTPS-based communication

- Authentication and authorization support
- Controlled data access

Connection-Level vs Step-Level Configuration

- **Connection-level configuration** applies globally
- **Step-level configuration** applies only to specific scenario steps

This provides flexibility and security control.

Data Connection Configuration

HTTP Methods

Supported methods include:

- GET (retrieve data)
- POST (send data)
- PUT and DELETE (update or remove data)

Headers and Query Parameters

Headers manage:

- Authentication tokens
- Content types

Query parameters pass filters and request-specific values.

Payload and JSON Handling

Payloads allow structured data exchange using JSON, enabling:

- Reliable data parsing
- Schema validation

Response and Error Variables

Responses are stored in variables for:

- Follow-up actions
- Error handling
- User feedback

Advanced Scenario Logic

Skill Elements

Skills represent reusable logic blocks that simplify complex workflows.

Global and Assign Elements

These elements manage:

- Variable storage
- Data sharing across steps

Action Elements (JavaScript ES5)

Custom logic can be written using JavaScript for:

- Data processing
- Conditional logic

Performance and Safety Constraints

Execution limits ensure:

- System stability
- Prevention of infinite loops
- Safe resource usage