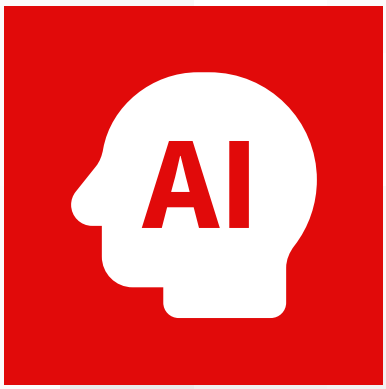
SEEBURGER

Solution Brief | SEEBURGER Capability Overview and Architectural Positioning

Agentic AI Capabilities within SEEBURGER BIS



Business challenges in modern integration landscapes

Modern integration scenarios increasingly require decisions that cannot be predefined. Processes span heterogeneous systems, involve unstructured information and rely on contextual interpretation. Traditional workflow automation reaches its limits when integrations depend on reasoning, multi-step decisions or dynamic next steps that change at runtime.

SEEBURGER extends deterministic orchestration with agentic AI capabilities directly within the Integrator Workspace (IWS) as part of the SEEBURGER Business Integration Suite (BIS). An AI Agent can reason over input, consult Large Language Model (LLM) providers and invoke tools such as subflows or external Model Context Protocol (MCP) services, all under governance and observability. This enables context-aware automation for scenarios that previously required manual decisions, while maintaining full control, transparency and operational safety.

Agentic AI capabilities within SEEBURGER BIS

Agentic capabilities extend integration workflows with dynamic decision-making and multi-step reasoning. They support scenarios in which contextual logic, tool-assisted processing, or iterative problem solving is required to complete a task.

Within SEEBURGER BIS, the AI agent component provides these agentic capabilities in a controlled and auditable runtime environment. It complements existing orchestration and integration logic by enabling LLM-driven analysis and autonomous decision sequences directly inside a workflow.

The agent operates iteratively: It evaluates the current task, performs one or more reasoning steps, optionally invokes tools, and continues until the task is completed or a termination condition is met. All activities run within the BIS execution environment, ensuring transparency, traceability, and operational control across every stage of reasoning and tool interaction.

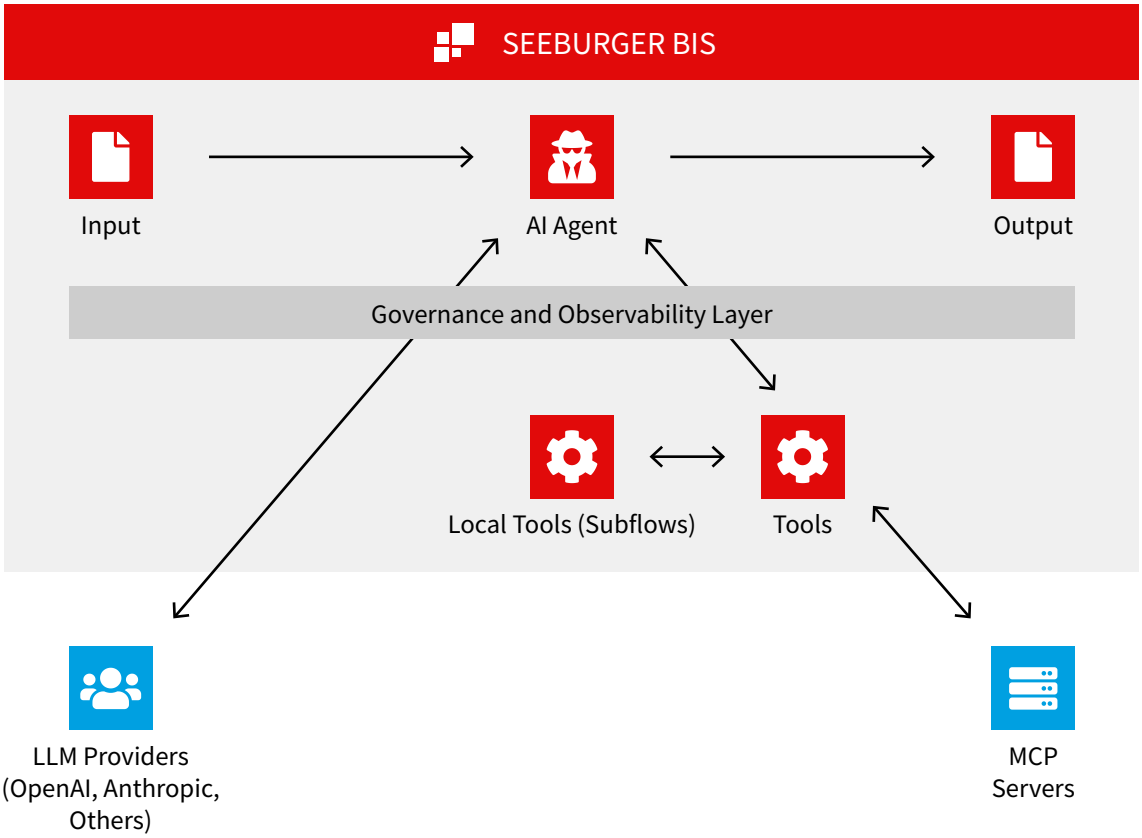


Figure 1: AI agent interaction model with LLM providers and tools, governed by execution controls

Configuring the AI agent component

The AI agent component is available as part of the platform's AI-enabled workflow elements. It can be placed at any point in an orchestration flow where autonomous reasoning or multi-step processing is needed.

Once added to a flow, the component exposes three configuration areas:

Chat Model:

Defines the LLM provider, model, output structure, and optional parameters that influence reasoning behavior.

General:

Controls execution aspects such as iteration limits, time-outs, and prompt definitions.

Agent Tools:

Allows the agent to access reusable logic encapsulated in subflows or external tools.

This structure separates model configuration, execution behavior, and tool integration, making the setup transparent and maintainable. It ensures that agentic logic aligns with platform governance, operational boundaries, and established integration patterns.

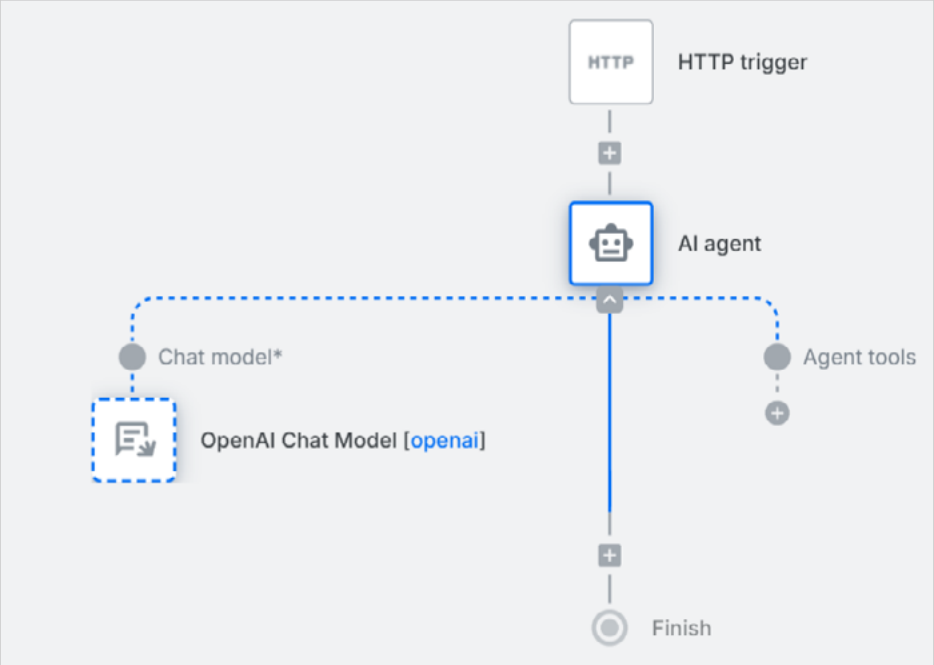


Figure 2: AI agent component integrated as a workflow component within the orchestration environment

Chat Model

The Chat Model defines how the agent interacts with an LLM. It specifies the selected provider and model, the applied parameters, and the expected output structure. Within BIS, the Chat Model determines the reasoning behavior used during agent execution and provides predictable, controlled handling of LLM outputs.

The configuration supports:

+ **Provider and model selection:**
  The agent can connect to different LLM providers. Model names, base URLs, and provider-specific capabilities are derived from the selected provider.
+ **Credential assignment:**
  Access to the LLM endpoint is governed by account-based credential management, ensuring secure and compliant model usage.
+ **Optional model parameters:**
  Depending on the provider, additional controls such as temperature, timeouts, reasoning effort, or token limits can be applied to refine the agent's behavior.
+ **Response formats** including:
  - **Text** for unstructured outputs
  - **JSON** for structured responses following a prompt-defined schema
  - **Structured JSON** validated against an uploaded JSON schema

These settings allow the agent to operate with consistency, governance, and provider neutrality. They ensure that reasoning steps and output formats align with workflow requirements and downstream processing.

General settings

The General section controls execution behavior, operational limits, and the prompts that determine how the agent interprets its task.

The agent supports:

+ **Iteration limits**, which prevent unbounded reasoning cycles. Each iteration includes a reasoning step and, if configured, a tool invocation. When no tools are available, the agent performs a single reasoning cycle similar to a standard LLM call.
+ **Timeout settings**, defining the maximum wait time for model responses during execution.
+ **System prompts**, which establish the agent's role, operational constraints, and overall behavior within the workflow.
+ **Chat input**, which provides task-specific instructions and defines the objective for a given workflow run.

These controls ensure that the agent behaves deterministically and respects operational boundaries. They also allow the agent's reasoning behavior to be aligned with integration logic, compliance requirements, and runtime expectations.

Agent Tools configuration

The Agent Tools section enables the AI Agent to use additional processing capabilities during execution. These capabilities are provided through subflows created within the same Integrator Workspace project. Each selected subflow becomes available to the agent as a callable tool.

The configuration consists of three steps:

01

Selecting one or more subflows to expose as tools. Subflows must exist in the current project to ensure consistency and access control.

02

Defining the tool specification, which describes the function and intended purpose of the subflow. This specification guides the agent's reasoning process when deciding whether and how to invoke the tool.

03

Providing the input definition, outlining the parameters that the tool expects. These definitions ensure that the agent issues complete and valid requests during execution.

Key model- and tool-related settings are also reflected visually in the flow graph, providing a clear and navigable representation of how agentic reasoning integrates with workflow logic. This supports transparency and maintainability across complex workflows.

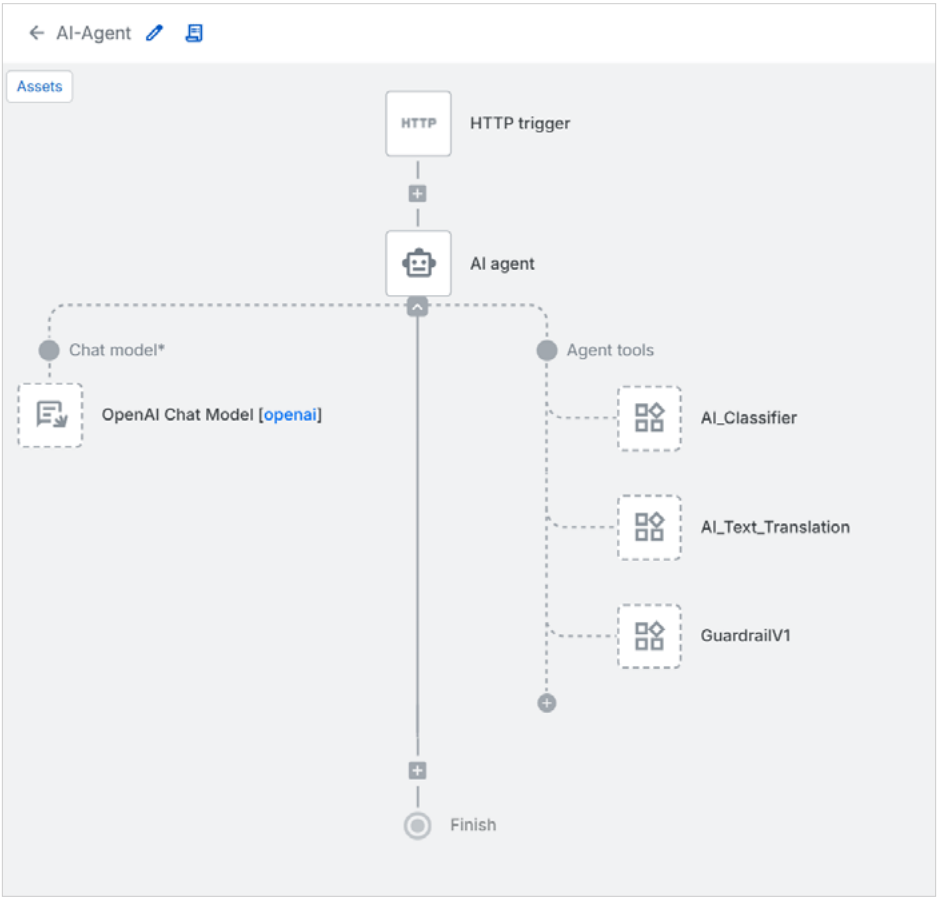


Figure 3: AI agent orchestrating model interaction and tool invocation within a workflow

Extending an agent with additional tools

The agent's capabilities can be expanded at any time by attaching additional tools. When multiple tools are available, the agent autonomously determines which tool to call and how to parameterize it based on the current task, system prompt, and reasoning outcome.

In a typical scenario, the agent may:

- perform several reasoning cycles,
- classify incoming information,
- invoke a transformation or analysis tool depending on the prompt,
- and continue iterating until the defined task is fully resolved.

Each tool invocation includes the parameters provided to the subflow and the result returned. These details appear in the execution trace, offering full transparency into the agent's decision-making process.

The final output, returned through the component's output interface, may include classification results, enrichment data, transformed structures, or other processed information. This structured transparency ensures traceability and supports both troubleshooting and optimization of agent behavior.

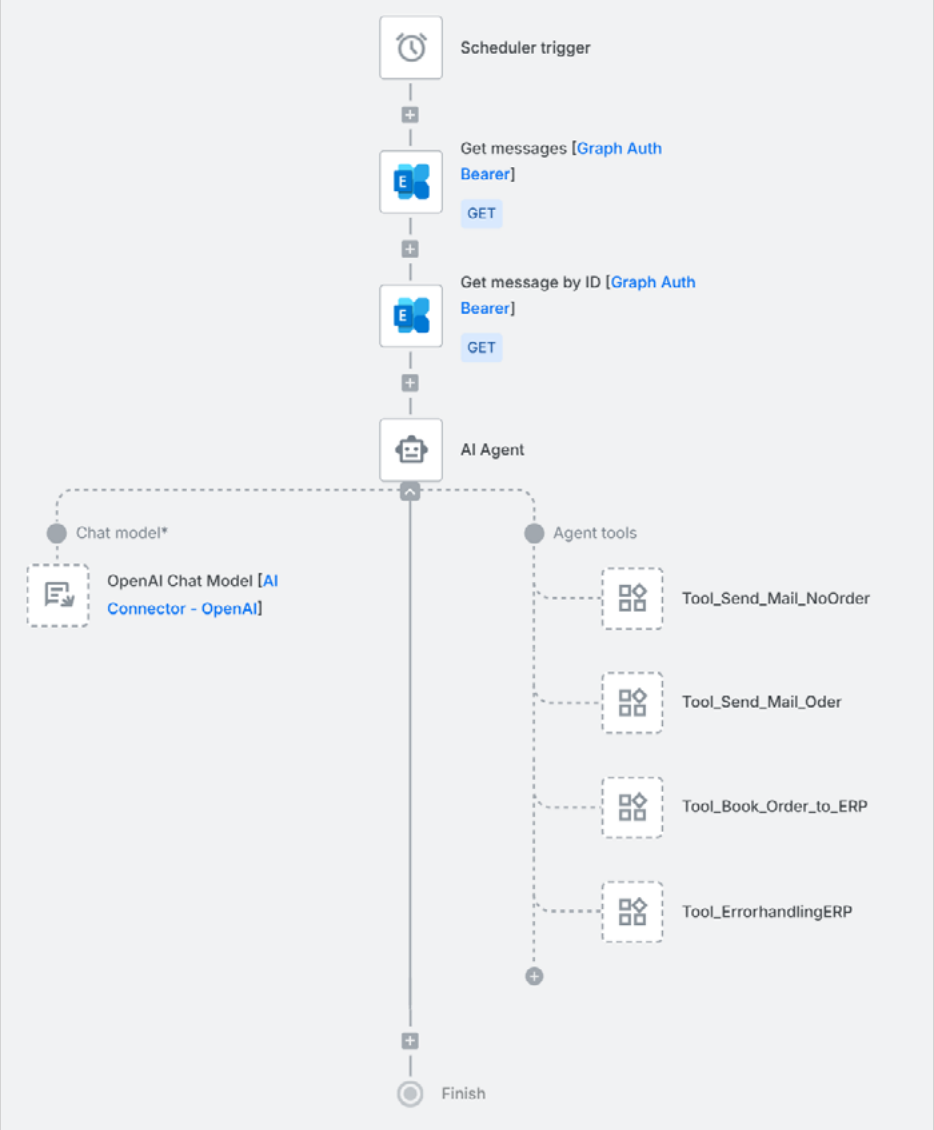


Figure 4: AI agent with multiple attached tools (subflows)

Combined execution example

When multiple tools are available, the agent determines during execution which tool to invoke and how to parameterize it based on the system prompt and the reasoning outcome of each iteration. This enables flexible, context-aware processing within a workflow.

In the example scenario, the agent performs several reasoning cycles to complete the task. During execution it:

1. **Classifies** the incoming content using a dedicated classification tool.
2. **Transforms** or enriches the data using either a translation or transformation tool, depending on the prompt and the detected context.
3. **Continues** iterating, invoking additional tools as needed until the defined objective is reached.

Each tool invocation is captured in the execution trace, including:

+ the payload sent to the tool,
+ the applied parameters,
+ and the returned result.

This provides a complete and transparent record of the decision sequence. The final output exposed by the agent may include classification results, confidence indicators, enriched content, or structured data produced through the chained tool interactions.

Invoking and testing an AI Agent

Once configuration is complete, the AI Agent component can be executed using the Test Flow function within the Integrator Workspace. During a test run, the engine provides a structured and comprehensive view of the agent's behavior, including its final output, metadata, and full execution trace.

Output view

The **Output** view displays the final response returned by the agent. This response follows the format specified in the Chat Model configuration, such as text or JSON. It reflects the agent's reasoning steps and any tool interactions that contributed to producing the final result.

Metadata view

The **Metadata** view exposes technical details related to LLM execution. This includes:

- token usage,
- iteration counts,
- invoked tools,
- model-level performance indicators.

This information allows teams to understand the computational footprint of an agent run. It also supports operational monitoring, cost estimation, and optimization of reasoning or tool selection.

Details view

The **Details** view provides a complete execution trace of the agent within the Integrator Workspace engine. It captures:

- each reasoning iteration,
- every tool invoked by the agent,
- the input parameters passed to each tool,
- the results returned from each tool,
- and the final termination reason.

This detailed trace enables users to validate agent behavior, understand intermediate steps, and troubleshoot unexpected outcomes with full transparency.

Execution transparency and operational control

Because agentic reasoning introduces dynamic decision sequences, transparency and governance are essential. BIS ensures this through integrated runtime controls and consistent observability.

During execution, all agent activities are:

+ **fully logged** within the BIS workflow engine,
+ **auditable** through structured traces,
+ **deterministic** within defined iteration and timeout limits,
+ **aligned with platform governance** through controlled tool exposure and credential management.

The combination of output, metadata, and detailed execution tracing ensures that agent-driven workflows remain predictable, reviewable, and suitable for enterprise integration scenarios. This visibility supports iterative refinement and dependable operation in production environments.

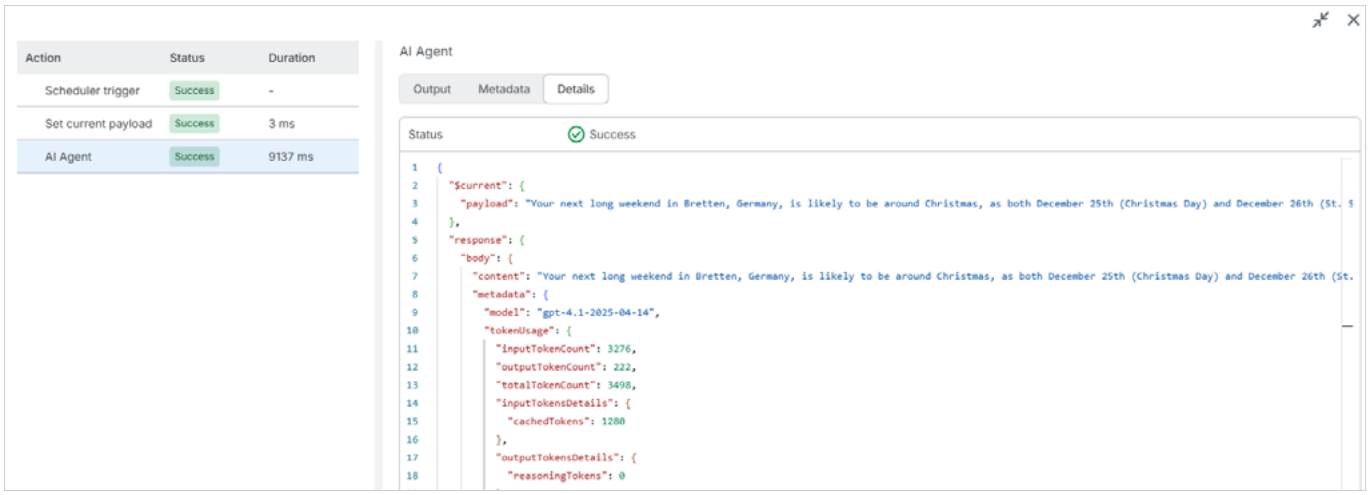


Figure 5: Execution trace of an AI agent run in the Test Flow view

MCP Client

The AI Agent can be extended with capabilities provided by external services through the built-in MCP Client. MCP enables the agent to retrieve structured information or perform specialized operations outside the BIS environment, while maintaining full control and traceability within the execution flow.

Connection options

The MCP Client supports two transport mechanisms:

- Server-Sent Events (SSE) for backwards compatibility
- Streamable HTTP, a modern protocol that enables continuous, token-by-token response streaming

Both options allow the agent to process structured results from external MCP servers as part of its reasoning cycle.

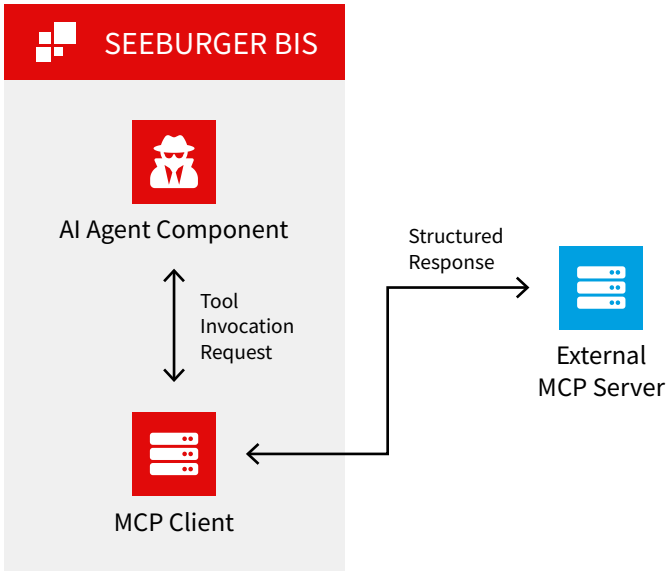


Figure 6: AI agent using the MCP Client to invoke external MCP servers

Authentication

To ensure secure access, the MCP Client supports multiple authentication methods, including:

- API Key
- Bearer Token
- OAuth 2.0
- Basic Authentication
- JWT

This flexibility allows MCP-based integrations to be added without modifying the surrounding flow logic.

Using MCP servers in agent runs

Once an MCP server is added to the workspace configuration, it becomes available as a callable resource during agent execution. Each external call is integrated into the standard execution trace, showing:

- when the MCP call was triggered,
- which parameters were passed,
- and what results were returned.

This ensures that external service interactions are as transparent and auditable as internal tool invocations, while expanding the range of capabilities available to the agent.

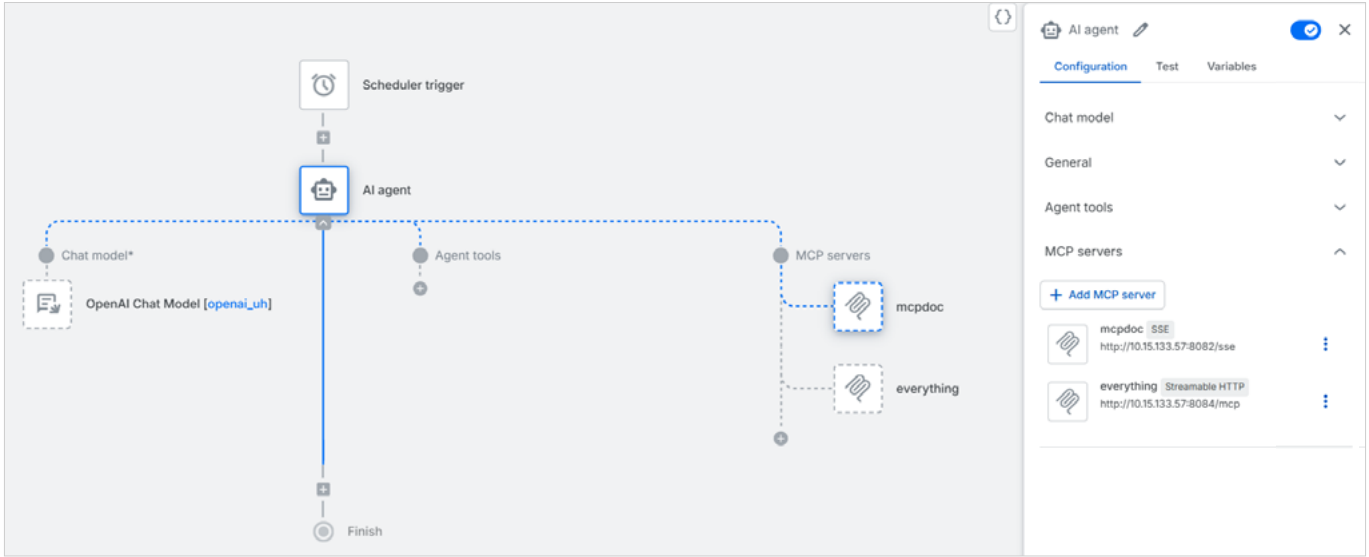


Figure 7: Calling MCP servers during agent execution.

Key benefits of agentic AI within the SEEBURGER BIS Platform

Higher automation coverage

Enables workflows that require interpretation, context evaluation or multi-step reasoning, reducing manual intervention in complex integration processes.

Consistent governance and transparency

All reasoning steps, tool invocations and external calls remain fully traceable within BIS, ensuring operational clarity and auditability.

Reusability through tool-based orchestration

Subflows and MCP-connected services can be reused across multiple agents, improving maintainability and reducing implementation effort.

Safe LLM integration

LLM-driven logic operates within BIS execution limits, timeouts and credential controls, providing a secure environment for AI-assisted automation.

Flexible extension capabilities

Agents can dynamically include subflows or external MCP services, allowing modular expansion without redesigning existing processes.

Typical use cases for agentic AI in integration scenarios

01 **Autonomous data quality assessment**

Agents can evaluate incoming payloads for completeness, consistency, or anomalies and decide whether to enrich, correct, or escalate the data before it enters downstream processes.

02 **Multi-source context aggregation**

Agents can gather information from several systems or services, interpret the combined context, and determine the appropriate next step – for example selecting the right process variant or assembling metadata for an end-to-end transaction.

03 **Intelligent process branching**

Instead of relying on static workflow rules, agents can analyze business inputs and dynamically decide which process branch, subflow, or API sequence to activate based on context, intent, or content semantics.

04 **Guided resolution of complex business tasks**

For cases that typically require human judgment – such as matching ambiguous records, interpreting mixed-format inputs, or selecting the most suitable transformation – agents can reason through alternatives and execute the resolution path autonomously.

05 **Runtime optimization and adaptation**

Agents can monitor execution context, detect deviations or unexpected patterns, and adjust workflow behavior at runtime. This includes selecting the most efficient service, adapting retry strategies, or invoking additional tools when needed to complete the task successfully.

Conclusion

Agentic AI transforms integration from predefined orchestration into adaptive, context-aware automation. By adding reasoning, autonomy, and controlled decision-making, organizations can automate scenarios that were previously impossible, from handling ambiguous data to selecting the next best action at runtime.

In SEEBURGER BIS, agentic capabilities extend existing integration patterns without sacrificing safety, governance, or transparency. They provide reusable logic, consistent behaviour, and dynamic execution that enhances today's operations while laying the groundwork for tomorrow's intelligent automation.

More importantly, the value is immediately actionable. Agentic AI can be applied to a wide range of integration challenges – independent of industry, architecture or existing workflows – giving enterprises a flexible and future-proof way to increase efficiency and reduce manual effort.

Agentic automation is not a trend.
It is the next strategic layer in modern integration.

> Contact us

Direct Contact

Ulf Persson
Senior Vice President Strategic Product Management and Analyst Relations

www.seeburger.com