



AI Agent Manager

Last updated: 03/03/2026

This content applies to the latest CD version of Cumulocity.

Specifications contained herein are subject to change and these changes will be reported in subsequent versions.

Copyright © 2026 Cumulocity GmbH.

The name Cumulocity GmbH and all Cumulocity GmbH product names are either trademarks or registered trademarks of Cumulocity GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

This software may include portions of third-party products. Third-party terms are set out in a 3rd-party-licenses file linked to or included with each installation package.

Table of Contents

Table of Contents	3
INTRODUCTION	5
WHAT IS AN AI AGENT?	5
WHAT YOU DO WITH AI AGENTS	5
GETTING STARTED BY CONFIGURING A GLOBAL PROVIDER	6
AGENTS	8
CUSTOM AGENTS	8
SYSTEM PROMPTS	8
VARIABLES	9
Defining variables in prompts	9
Providing variable values	9
Use cases for variables	10
SETTINGS AND ADVANCED SETTINGS	10
Common settings	10
Advanced settings	10
Provider-specific options	11
When to use settings	11
TOOLS	11
TEST	11
SUBSCRIBED AGENTS	12
WHAT ARE SUBSCRIBED AGENTS?	12
VIEWING SUBSCRIBED AGENTS	12
TESTING SUBSCRIBED AGENTS	13
OVERRULING SUBSCRIBED AGENTS	13
SUBSCRIBED AGENT VERSIONING	13
REMOVING SUBSCRIBED AGENTS	13
TEXT AND OBJECT AGENTS	13
TEXT AGENTS	14
OBJECT AGENTS	14
KEY DIFFERENCES	15
CHOOSING THE RIGHT TYPE	15
TESTING AGENT TYPES	16
CONVERTING BETWEEN TYPES	16
LOCAL PROVIDERS	16
WHAT ARE LOCAL PROVIDERS?	16
WHEN TO USE LOCAL PROVIDERS	16
GLOBAL PROVIDER VERSUS LOCAL PROVIDER	16
CONFIGURING A LOCAL PROVIDER	16
TESTING LOCAL PROVIDERS	17
MANAGING MULTIPLE LOCAL PROVIDERS	17
SECURITY CONSIDERATIONS	17
REMOVING A LOCAL PROVIDER	17
TROUBLESHOOTING LOCAL PROVIDERS	17
TOOLS AND MCP	18
TOOLS	18
WHAT ARE TOOLS?	18
USING AND TESTING BUILT-IN CUMULOCITY TOOLS	19
ASSIGNING TOOLS TO AGENTS	19
BEST PRACTICES	19
MCP SERVERS	19
MODEL CONTEXT PROTOCOL (MCP)	19
CONFIGURING MCP SERVERS	20
CREATING CUSTOM MCP SERVERS	21
BEST PRACTICES	21

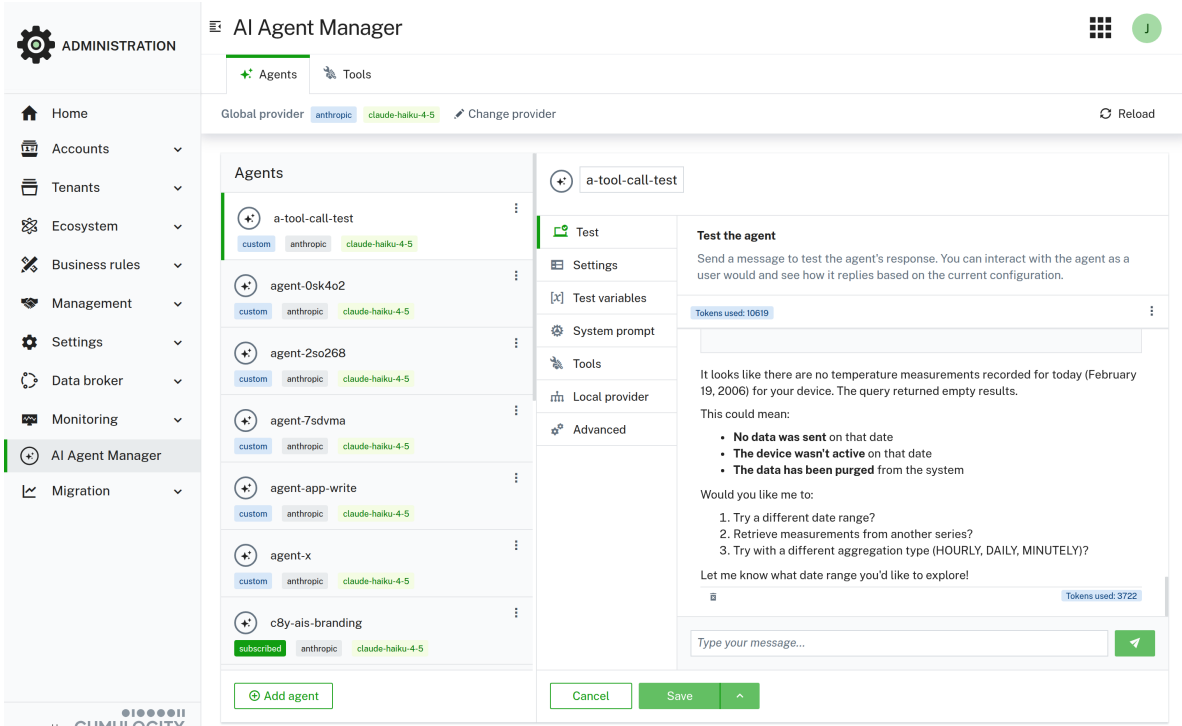
REST API	22
API OVERVIEW	22
INTERACTING WITH AGENTS	22
STREAMING	23

INTRODUCTION

FEATURE PREVIEW

The feature is currently in public preview and might change or is not feature complete. While we try to keep changes as low as possible, breaking changes and feature removals happen. To enable the feature, open the right drawer (by clicking on your username initials) in the **Administration** application and select **Manage preview features**. Then activate the toggle next to **AI Agent Manager**.

The AI Agent Manager enables you to create and manage AI agents—intelligent assistants that help users interact with your IoT data and systems through natural language conversations. Instead of building custom AI integrations from scratch, you configure pre-built agents that understand your specific use cases and have access to the tools required to be helpful.



WHAT IS AN AI AGENT?

An AI agent is a configured AI assistant with a specific purpose and behavior. Each agent consists of:

- **System prompt:** Defines the agent's role, personality, and expertise (for example, "You are a factory monitoring assistant").
- **Tools:** Actions the agent performs, such as querying device data, triggering operations, or accessing custom services.
- **AI provider:** The underlying AI model (such as Claude, GPT, or others) that powers the agent.

Once configured, users interact with agents through natural language without needing to know technical details about APIs or system prompts.

WHAT YOU DO WITH AI AGENTS

- **Create specialized assistants:** Configure agents for specific use cases like equipment monitoring, predictive maintenance analysis, or device troubleshooting. Each agent has its own expertise and access to relevant data.
- **Add tools and capabilities:** Connect agents to Cumulocity data through built-in tools or extend them with custom tools via MCP (Model Context Protocol) servers. This allows agents to query device data, retrieve measurements, or trigger actions.

- **Test and refine:** Use the built-in test interface to interact with your agents and see how they respond. The test interface shows you exactly what tools the agent is calling and how it reasons about your questions.
- **Provide conversational interfaces:** Once configured, agents maintain conversations with context, using variables to personalize responses and access to tools to fetch real-time data.
- **Manage AI providers:** Configure which AI provider and model to use globally or for specific agents. The system supports multiple providers including OpenAI, Anthropic, Google, and others, abstracting away the complexity of working with different APIs.

GETTING STARTED BY CONFIGURING A GLOBAL PROVIDER

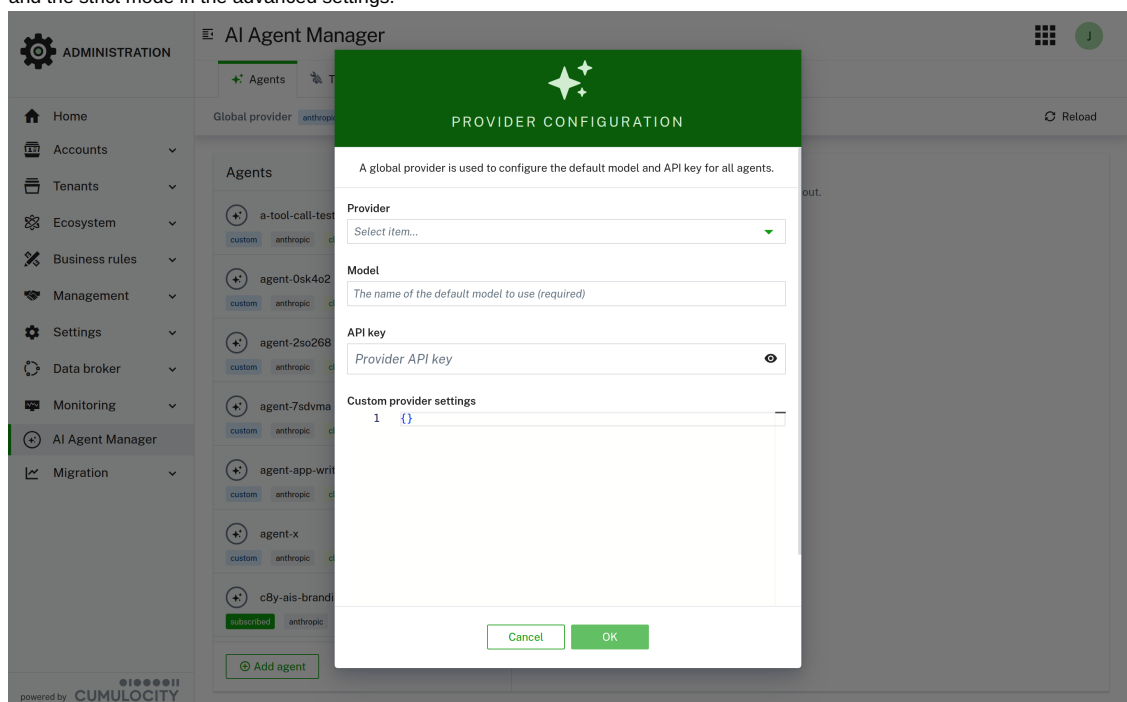
! IMPORTANT

Without a global LLM provider configured, the Cumulocity platform is never connected to any external Large Language Model provider, even if the list already shows subscribed agents.

Start by configuring a global provider and model. A global provider is the LLM provider and model that is used by default on each agent interaction as long as the agent does not define another provider or model in its local provider configuration. Use the AI Agent Manager UI in the **Administration** application:

1. Open the AI Agent Manager main view. You see a list of agents provided in your instance.
2. In the toolbar, click **Add global provider**.
3. In the modal, configure your provider. The tested providers are:
 - Anthropic
 - OpenAI
 - Google Gemini

Additional providers are supported but not tested by the Cumulocity team. They work without guarantee. The same applies to open source models based on the OpenAI API, which are supported by using the OpenAI provider and changing the baseURL and the strict mode in the advanced settings.



4. Define the model to use.
5. Add an API key provided by your provider. This is securely stored inside the platform and cannot be read afterwards.
6. Add advanced provider-specific settings if necessary. The advanced settings editor accepts JSON and validates via a JSON schema.
7. Save the provider.

The next step is to validate your provider. The easiest way is to create a new agent:

1. Click **Add agent** at the bottom of the list.
2. Select **Test** and write "Hello world".
3. The agent responds with a hello message.

If successful, your provider is correctly configured, and you can start to use and test AI agents. Next, learn more about subscribed agents, object agents, local providers, and MCP tools:

- Learn what [subscribed agents](#) are and how you can align them.
- Understand the difference between an [object and a text agent](#).
- See how you can leverage [local providers](#) to overwrite the provider or model of a global provider.
- Understand the available [tools and how you can extend them with MCP](#).
- You can also manage agents programmatically [via the REST API](#) for integration into your own applications and workflows.

AGENTS

Agents are the basic building block of any agentic workflow. The AI Agent Manager simplifies the creation of agents by providing a UI that lets you quickly define an agent with a system prompt that then, on its own, calls tools. This quick starts agentic workflows in a couple of minutes, not hours.

CUSTOM AGENTS

To create a custom agent, navigate to **Administration > AI Agent Manager** and click **Add agent**. Select an agent name and the type of agent. This introduction only describes text agents. For details about object agents, refer to [Text and object agents](#). When selecting a name, remember that naming conflicts can occur with subscribed agents provided from applications or plugins. It is good practice to use a prefix other than `c8y-`, as this is the default prefix used by the platform.

Once the custom agent is created, you align it to your needs using the following tabs:

- **Test:** Test your agent directly in the AI Agent Manager.
- **Settings:** Allows to set settings like the maximum output tokens or temperature.
- **Test variables:** Set variable values for your test.
- **System prompt:** The system prompt of the agent. You align it and then test the changes. The system prompt persists only when you save it.
- **Tools:** Assign tools to your agent.
- **Local provider:** Allows to set different LLM provider or models for one agent ([\(read more\)\[\(/ai/agents/#local-providers\)\]](#)).
- **Advanced:** Enables advanced settings in JSON format.

SYSTEM PROMPTS

The system prompt defines your agent's behavior, expertise, and personality. It is the foundational instruction that shapes how the agent interprets user questions and formulates responses. Unlike user messages that change with each conversation, the system prompt remains constant and guides the agent throughout all interactions.

The screenshot displays the 'AI Agent Manager' interface. On the left is a sidebar with navigation options: Home, Accounts, Tenants, Ecosystem, Business rules, Management, Settings, Data broker, Monitoring, AI Agent Manager (selected), and Migration. The main area is titled 'AI Agent Manager' and has tabs for 'Agents' and 'Tools'. Under the 'Agents' tab, a list of agents is shown, including 'a-tool-call-test', 'agent-0sk4o2', 'agent-2so268', 'agent-7sdvma', 'agent-app-write' (selected), 'agent-x', and 'c8y-ais-branding'. Each agent entry shows its type (custom or subscribed) and the provider (anthropic or claudia-haiku-4-5). The 'agent-app-write' agent is selected, and its configuration is shown in a modal. The modal has tabs for 'Test', 'Settings', 'Test variables', 'System prompt' (selected), 'Tools', 'Local provider', and 'Advanced'. The 'System prompt' tab contains a text area with the following content:

```

1 You are responsible to create an IoT app. Therefore first
clone the 'cockpit' application by using the
cumulocity-clone-application tool. Then align the index.
html to the need of the user. Select an app named, based on
the user input.
2
3 ## Rules for building the app
4 1. always clone the 'cockpit' app!
5 2. On the first change, you should not try to align the
index.html. Just overwrite it with an new app.
6 3. You are authenticated via cookie auth. If not (401
returned from the cumulocity api), redirect the user to /
apps/login
7
8 ## Rules for file changes
9 If you want to change a file and not overwrite it
completely, always read it before to ensure the status
wasn't changed before.

```

At the bottom of the modal are buttons for 'Cancel', 'Save', and an up arrow.

What to include in a system prompt:

- Write clear and specific instructions about the agent's role and purpose. For example, "You are a device troubleshooting assistant for industrial IoT equipment" is more effective than "You are helpful."
- Define the agent's tone and communication style. Specify whether responses are formal, conversational, technical, or simplified

for non-technical users.

- Set boundaries and limitations. Explicitly state what the agent does not do or what topics it avoids. For example, "You do not provide financial advice or make purchasing decisions."
- Include domain knowledge and context. Add relevant background information about your IoT environment, device types, or specific terminology the agent needs to understand.
- Specify output format preferences. Indicate whether responses are concise bullet points, detailed explanations, or follow a specific structure.

Do's:

- Be specific and concrete rather than vague or general.
- Test different prompt variations to find what works best for your use case.
- Include examples of desired behavior directly in the prompt.
- Update the system prompt based on observed agent behavior.
- Keep the prompt focused on a single, clear purpose.
- Use the agent's perspective (write as "You are..." not "The agent is...").

Don'ts:

- Avoid contradictory instructions that confuse the agent.
- Do not make the prompt excessively long (generally stay under 2000 words).
- Avoid assumptions about what the agent "knows" - be explicit.
- Do not include user-specific information that changes per interaction (use variables instead).
- Avoid overly complex or nested conditional logic.
- Do not use ambiguous language that has multiple interpretations.

VARIABLES

Variables allow you to inject dynamic data into your system prompt or user prompts at runtime. Instead of hardcoding specific values, you define placeholders that get replaced with actual values when the agent is called.

Defining variables in prompts

Use double curly brackets to define variables: `{{variableName}}`. You place variables anywhere in the system prompt or in API calls.

Example system prompt with variables:

```
You are a monitoring assistant for factory {{factoryId}}. When users ask about equipment, focus on devices in the {{location}} area. Current shift manager is {{shiftManager}}.
```

Providing variable values

When testing in the AI Agent Manager, use the **Test variables** tab to set values for your variables before testing the agent. You simply provide the variable as JSON where the key is the variable name and the value is the value you want the variable to be. For the above example you would need to add to the **Test variables** tab the following JSON:

```
{
  "factoryId": "FAC-001",
  "location": "Building A",
  "shiftManager": "John Smith"
}
```

When calling the agent via REST API, provide variables in the request body:

```
{
  "variables": {
    "factoryId": "FAC-001",
    "location": "Building A",
    "shiftManager": "John Smith"
  },
  "prompt": "What is the status of equipment in {{location}}?"
}
```

Use cases for variables

- Personalizing responses with user-specific information (names, roles, preferences).
- Contextualizing agents for different locations, facilities, or departments.
- Injecting current state information that changes frequently.
- Reusing the same agent configuration across multiple contexts.

Variables make agents flexible and reusable without requiring multiple agent configurations for similar use cases.

! IMPORTANT

System prompts and variables are vulnerable to prompt injection attacks. Always sanitize and validate any input used in prompts or variables, as the AI Agent Manager does not provide automatic protection. Learn more about prompt injection risks and mitigation strategies on the [OWASP website](#).

SETTINGS AND ADVANCED SETTINGS

The settings allow you to fine-tune the agent's behavior using parameters from the Vercel AI SDK. These settings control aspects like response randomness, length limits, and provider-specific features.

Common settings

There are common settings that you can set in the **Setting** tab:

Parameter	Range	Description
maxOutput Tokens	Number (min: 1)	Sets the maximum length of the response in tokens. Use this to enforce concise responses or prevent excessively long outputs.
temperature	0.0 to 1.0	Controls response randomness. Higher values make output more random, lower values more deterministic. Note: Cannot be used simultaneously with topP.
topP	0.0 to 1.0	Nucleus sampling parameter. Only tokens with top probability mass are considered (e.g., 0.1 = top 10%). Note: Cannot be used simultaneously with temperature.
topK	Number (min: 1)	Limits sampling to the top K options. For example, 40 means only the top 40 token options are considered.
presencePenalty	-1.0 to 1.0	Encourages the model to introduce new topics. Positive values make the agent less likely to repeat topics already mentioned.
frequencyPenalty	-1.0 to 1.0	Reduces repetition of tokens based on their frequency in the response. Positive values discourage repeated words.
seed	Number (min: 0)	Sets a seed for reproducible results. Using the same seed with the same inputs produces consistent outputs.
maxRetries	Number (min: 0)	Number of times to retry the request on failure. For example, 2 means the system will retry up to 2 times.
stopSequences	Array of strings	Sequences that, when generated, will stop the response. For example, "END" or "\n\n". Only available for text agents.

Advanced settings

In the **Advanced settings** tab a user can specify more options, which don't have yet an UI form. The options are based on the used Vercel AI SDK. For a complete list of available parameters and provider-specific options, refer to the [Vercel AI SDK documentation](#).

Provider-specific options

Some AI providers support additional features configured through the `providerOptions` field. For example, Anthropic's extended thinking mode:

```
{
  "anthropic": {
    "thinking": {
      "type": "enabled",
      "budgetTokens": 12000
    }
  }
}
```

When to use settings

- Adjust temperature when responses are too random or too rigid.
- Set maxTokens to control costs or enforce response brevity.
- Use penalties to reduce repetitive language in longer conversations.
- Enable provider-specific features for specialized capabilities.

Test different settings to find the optimal configuration for your use case.

TOOLS

Tools extend your agent's capabilities by allowing it to access data and perform actions beyond generating text. In the **Tools** tab, you assign tools from configured MCP servers to your agent.

To assign tools to an agent:

1. Navigate to the **Tools** tab in the agent configuration.
2. Browse the list of available tools from configured MCP servers.
3. Select the tools your agent needs to answer user questions.
4. Click **Save** to update the agent configuration.

The agent automatically determines when to call assigned tools based on user questions and tool descriptions. For example, if you assign a "get device measurements" tool, the agent calls it when users ask about current temperature or sensor readings.

For detailed information about tools, configuring MCP servers, and how agents use tools, refer to [Tools and MCP servers](#).

INFO

Object agents cannot use custom tools. They use tools internally to structure responses according to their defined schema.

TEST

The **Test** tab provides an interactive interface to test your agent directly in the AI Agent Manager. This allows you to validate the agent's behavior, system prompt, and tool usage before deploying it in production.

To test an agent:

1. Open the agent configuration in the AI Agent Manager.
2. Navigate to the **Test** tab.
3. If your agent uses variables, set them in the **Test variables** tab first.
4. Enter a prompt in the chat interface.
5. Review the agent's response.

The test interface maintains conversation context, allowing you to have multi-turn conversations and verify the agent remembers previous messages.

SUBSCRIBED AGENTS

Subscribed agents are AI agents that are provided by installed applications in your Cumulocity tenant. These agents are automatically available in the AI Agent Manager once the providing application or plugin is deployed.

WHAT ARE SUBSCRIBED AGENTS?

When developers build applications or plugins that include AI functionality, they define agents to be exported. These agents are then “subscribed” to your tenant and appear in the AI Agent Manager list.

Subscribed agents come with predefined:

- System prompts that define their behavior and expertise.
- Tool configurations that allow them to interact with specific Cumulocity data or services.

Subscribed agents properties:

- **Pre-configured:** Subscribed agents are fully configured by the application provider. You do not need to define system prompts or tools.
- **Application-specific:** Each subscribed agent is designed for a specific use case within an application (for example, a device troubleshooting agent for a specific device management app).
- **Require global provider:** Subscribed agents use your configured global provider unless they specify a local provider. Without a global provider configured, subscribed agents remain inactive.
- **Read-only configuration:** You cannot modify the system prompt or tools of subscribed agents. However, you view their configuration to understand their capabilities. You also overwrite the subscribed agent with a custom agent. Custom agents with the same name as the subscribed one are preferred.
- **Automatic updates:** When the providing application updates the agent definition, changes appear automatically in your AI Agent Manager as long as you haven't overwritten it.

VIEWING SUBSCRIBED AGENTS

To view subscribed agents:

1. Navigate to **Administration > AI Agent Manager**.
2. In the agents list, subscribed agents display with a badge indicating their source application.
3. Click on a subscribed agent to view its details, including the system prompt and available tools.

The screenshot displays the AI Agent Manager interface. On the left is a sidebar with navigation options: ADMINISTRATION, Home, Accounts, Tenants, Ecosystem, Business rules, Management, Settings, Data broker, Monitoring, AI Agent Manager (selected), and Migration. The main panel shows the 'Agents' tab with a list of subscribed agents, all named 'c8y-html-widget', using the 'anthropic' provider and 'claude-haiku-4.5' model. A 'Clone agent' button is visible next to each entry. The right panel shows the configuration for the selected 'c8y-html-widget' agent. It includes a 'System prompt' section with a detailed instruction set for analyzing user requests, verifying APIs, and coding. The prompt is as follows:

```

1. **Analyze the user request**
2. - Extract specific data requirements
3. - Identify visualization needs
4. - Note any context dependencies
5.
6. 2. **API Verification**
7. - Use the "cumulocity-api-request" tool to verify needed APIs
8. - Document the exact API endpoints, parameters, and expected responses
9.
10. 3. **Coding**
11. - Put out one code block with only the code, wrap it in a <c8y-code-extract> block
12. - IMPORTANT: And no other text or markdown formatting inside the <c8y-code-extract> block
13. - build the widget code using the following rules: You

```

At the bottom of the configuration panel are buttons for 'Cancel', 'Save', and an upward arrow.

TESTING SUBSCRIBED AGENTS

You test subscribed agents the same way as custom agents:

1. Open the subscribed agent in the AI Agent Manager.
2. Navigate to the **Test** tab.
3. Enter a prompt and observe the agent's response.

OVERRULING SUBSCRIBED AGENTS

While you cannot change the core configuration of subscribed agents, you align them by overruling the agent. Click the three dots next to the subscribed agent and select **Clone agent**. The cloned agent then aligns to your needs, and all subscribed agents by any app only use this new custom agent.

SUBSCRIBED AGENT VERSIONING

While agents in general are not versioned, subscribed agents are versioned. They are provided by a custom or subscribed plugin that gets versioned, so the agents also exist in different versions. The AI Agent Manager shows the "latest" version of the plugin-agent. However, a custom application might use a different version. If the agent is overruled, always the custom user-defined agent is used.

REMOVING SUBSCRIBED AGENTS

Subscribed agents are removed automatically when you uninstall or remove the providing application. You cannot manually delete subscribed agents while their source application remains installed. If the source application is subscribed, you need to unsubscribe this application to uninstall the agent.

TEXT AND OBJECT AGENTS

The AI Agent Manager supports two base types of agents: text agents and object agents. Understanding the difference helps you choose the right type for your use case.

TEXT AGENTS

Text agents return natural language responses as plain text. They are designed for conversational interactions where the agent provides explanations, answers, or guidance in a human-readable format.

Use cases:

- Conversational interfaces where users ask questions in natural language.
- Explanations and guidance for device troubleshooting.
- General-purpose AI assistants that interact through chat.
- Generating reports or summaries in text format.

Response format:

By default, text agents return plain text:

```
The current temperature is 23.5°C and the humidity level is at 45%.
```

Add the `?fullResponse=true` query parameter to receive a JSON response with additional metadata, including tool calls, reasoning steps, and usage statistics.

API endpoint:

```
POST /service/ai/agent/text/{agent-name}
```

OBJECT AGENTS

Object agents return structured data in JSON format according to a predefined schema. They are designed for programmatic integrations where the response must follow a specific structure.

Use cases:

- APIs that require structured responses.
- Data extraction where specific fields must be populated.
- Integration with other systems that expect JSON.
- Form filling or data validation workflows.

Response format:

Object agents always return JSON that matches the defined schema:

```
{
  "temperature": 23.5,
  "humidity": 45,
  "status": "normal"
}
```

Schema definition:

When creating an object agent, define the expected response structure using JSON schema:

```
{
  "type": "object",
  "properties": {
    "temperature": {
      "type": "number",
      "description": "Current temperature in Celsius"
    },
    "humidity": {
      "type": "number",
      "description": "Current humidity percentage"
    },
    "status": {
      "type": "string",
      "enum": ["normal", "warning", "critical"]
    }
  },
  "required": ["temperature", "humidity", "status"]
}
```

The agent uses this schema to structure its response, ensuring consistent output format.

API endpoint:

POST /service/ai/agent/object/{agent-name}

UI support:

When creating an object agent a new **Schema** tab is shown. In this view you can define the correct schema and a validation will check, if you are following the right JSON Schema standard.

KEY DIFFERENCES

Aspect	Text agents	Object agents
Response format	Plain text (or JSON with <code>?fullResponse=true</code>)	Always JSON
Schema	Not required	Requires JSON schema
Tools	Supports custom tools	Cannot use additional tools (uses tools internally for structuring)
Use case	Conversational AI	Programmatic integration
Flexibility	High - can adapt response format	Low - follows strict schema

CHOOSING THE RIGHT TYPE

Choose text agents when:

- Building conversational interfaces or chatbots.
- Users need natural language explanations.
- Response format varies based on context.
- You want to use custom tools for data access.

Choose object agents when:

- Integrating with APIs or other systems.
- Response must follow a strict structure.
- Extracting specific data fields from user input.
- Building forms or structured data collection.

TESTING AGENT TYPES

You test both agent types in the AI Agent Manager:

1. Navigate to **Administration > AI Agent Manager**.
2. Create or open an agent.
3. The agent type is selected during creation and cannot be changed later.
4. For object agents, define the JSON schema in the configuration.
5. Use the **Test** tab to verify responses match your expectations.

CONVERTING BETWEEN TYPES

You cannot convert an existing agent from one type to another. To change the agent type, create a new agent with the desired type and copy the relevant configuration.

LOCAL PROVIDERS

Local providers allow you to configure agent-specific AI provider and model settings that override the global provider configuration. This enables you to use different AI models or providers for different agents based on their specific requirements.

WHAT ARE LOCAL PROVIDERS?

A local provider is an agent-specific configuration that defines:

- Which AI provider to use (for example, OpenAI, Anthropic, Google).
- Which model to use (for example, gpt-4, claude-3-7-sonnet).
- Provider-specific settings like API keys, base URLs, or custom parameters.

When an agent has a local provider configured, it uses those settings instead of the global provider settings.

WHEN TO USE LOCAL PROVIDERS

Consider the following:

- **Different model requirements:** Some use cases benefit from specific models. For example, use a faster, cheaper model for simple queries and a more powerful model for complex reasoning tasks.
- **Cost optimization:** Route less critical agents to more cost-effective models while keeping important agents on premium models.
- **Provider-specific features:** Access features only available from certain providers, such as extended thinking modes or specialized capabilities.
- **Testing and comparison:** Test different models side-by-side by creating multiple agents with different local providers.
- **Separate billing:** Use different API keys for different agents to track usage or allocate costs to different departments.

GLOBAL PROVIDER VERSUS LOCAL PROVIDER

Aspect	Global provider	Local provider
Scope	All agents without local providers	Single agent only
Configuration location	AI Agent Manager settings	Individual agent settings
Fallback	Used when no local provider is defined	Overrides global provider
Use case	Default for most agents	Special requirements

CONFIGURING A LOCAL PROVIDER

To configure a local provider for an agent:

1. Navigate to **Administration > AI Agent Manager**.
2. Open the agent you want to configure or create a new agent.
3. In the agent configuration, expand the **Local provider** tab.

In this view you can configure your local provider (only with JSON). Therefore (depending on the provider to use) you can define a new `provider`, `model` or `apiKey`. This settings are always merged with the "global provider" and therefore allow to e.g. only overwrite the model to be used.

For example to use an Open AI API like LLM hosted for example on an own infrastructure, the following configuration could be used:

```
{
  "provider": "openai",
  "model": "my-custom-gpt",
  "baseUrl": "https://your-custom-endpoint.com/v1",
  "strictMode": false
}
```

TESTING LOCAL PROVIDERS

After configuring a local provider:

1. Navigate to the **Test** tab of the agent.
2. Enter a test prompt.
3. Verify the response uses the local provider.

MANAGING MULTIPLE LOCAL PROVIDERS

You configure local providers independently for each agent. This allows you to:

- Use OpenAI for agent A, Anthropic for agent B, and Google for agent C.
- Test the same agent configuration with different models by creating duplicate agents with different local providers.
- Maintain separate API keys for different use cases or cost centers.

SECURITY CONSIDERATIONS

- **API keys:** Local provider API keys are stored securely in Cumulocity and cannot be read after configuration. Only users with appropriate permissions access local provider settings.
- **Access control:** Ensure only authorized users have permission to configure agents and local providers, as this grants access to external AI services.

REMOVING A LOCAL PROVIDER

To remove a local provider and revert to the global provider simply empty the JSON object.

TROUBLESHOOTING LOCAL PROVIDERS

- **Agent not responding:** Verify the API key is valid and the provider account has sufficient credits.
- **Different results than expected:** Check the model selection and advanced settings in the local provider configuration.
- **Provider not available:** Ensure the provider is supported by the AI Agent Manager. For the current list of supported providers, refer to the Vercel AI SDK documentation.

TOOLS AND MCP

Tools extend the capabilities of AI agents by allowing them to perform actions such as querying device data, retrieving measurements, or accessing custom services. The AI Agent Manager supports built-in Cumulocity tools and custom tools via Model Context Protocol (MCP) servers.

TOOLS

Tools are actions that AI agents perform to access data or execute operations. They enable agents to interact with Cumulocity and external systems beyond just generating text responses.

WHAT ARE TOOLS?

A tool is a function that an agent calls during a conversation to:

- Query device data or measurements.
- Retrieve information from Cumulocity APIs.
- Execute operations or commands.
- Access external services or databases.

When you ask an agent a question that requires data, the agent determines which tools to use, calls them with appropriate parameters, and incorporates the results into its response.

Example conversation:

User: "What is the current temperature of device 12345?"

Agent process:

1. Recognizes it needs device data.
2. Calls the "get device measurements" tool with device ID 12345.

3. Receives the measurement data.
4. Formulates a natural language response: "The current temperature is 23.5°C."

USING AND TESTING BUILT-IN CUMULOCITY TOOLS

The AI Agent Manager provides default tools for accessing Cumulocity data:

- **Device queries:** Search and retrieve device information.
- **Measurement retrieval:** Access current and historical measurements.
- **Alarm management:** Query alarms and their status.
- **Event access:** Retrieve events from devices.
- **Operation status:** Check operation execution status.

! IMPORTANT

These tools access and also change the data that the executing user has access to. Test such tools in a testing environment.

These tools are available automatically and require no additional configuration. Test these tools by accessing the **Tools** section of the AI Agent Manager. Open a tool, which displays the tool's description and parameters. Enter something into the chat box to test the tool. These tools always use the authorization of the current user and therefore cannot access more data than the user accesses. However, be aware that they change or delete data.

ASSIGNING TOOLS TO AGENTS

To enable an agent to use tools:

1. Open the agent configuration.
2. Navigate to the **Tools** tab.
3. Browse available tools.
4. Select the tools the agent needs.
5. Click **Save**.

The agent now accesses these tools during conversations when needed.

i INFO

Object agents cannot use custom tools. They use tools internally to structure their responses according to the defined schema.

BEST PRACTICES

- **Tool selection:** Assign only the tools an agent actually needs. Too many tools confuse the agent and increase processing time.
- **Performance:** Tools that take a long time to execute slow down agent responses. Optimize tool performance or use caching where appropriate.

MCP SERVERS

MODEL CONTEXT PROTOCOL (MCP)

The Model Context Protocol (MCP) is a standard for connecting AI agents to external tools and data sources. MCP servers expose

custom tools that extend agent capabilities beyond the built-in Cumulocity tools.

MCP server characteristics:

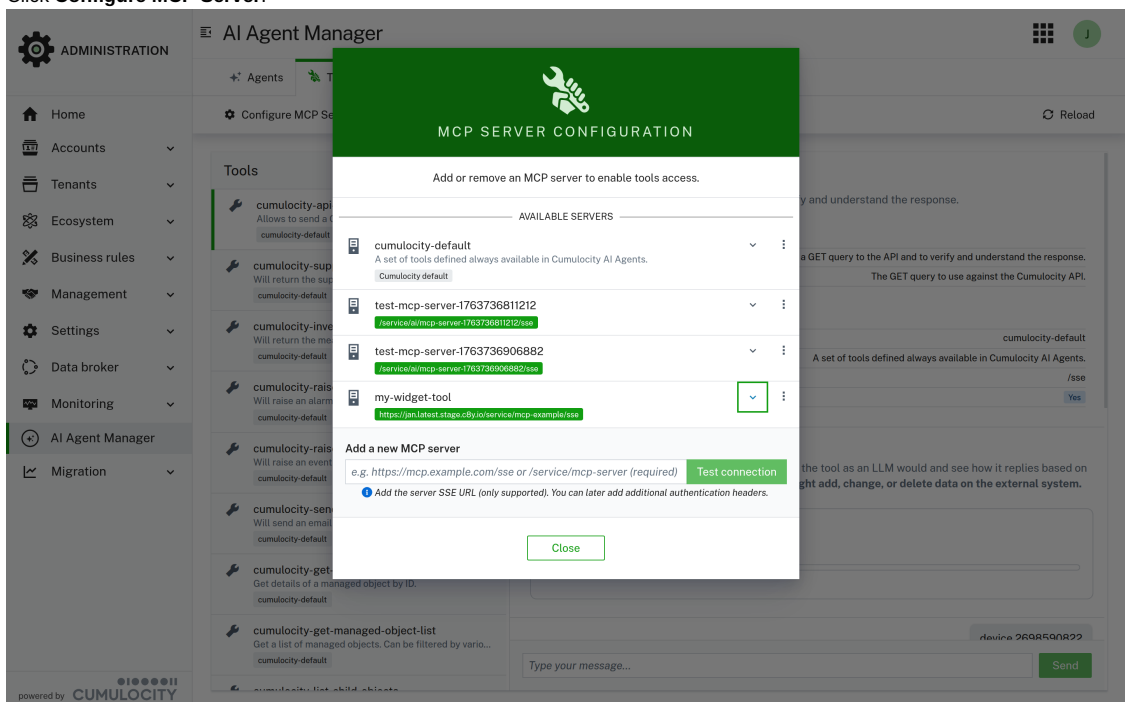
- Runs as a separate service accessible via SSE (Server-Sent Events).
- Exposes one or more tools that agents use.
- Defines tool schemas that describe parameters and functionality.
- Executes tool logic and returns results to the agent.

They are therefore perfect fits to extend the capabilities of Cumulocity agents, either by connecting an existing MCP server or by building your own.

CONFIGURING MCP SERVERS

To add an MCP server to the AI Agent Manager:

1. Navigate to **Administration > AI Agent Manager**.
2. Click **Tools** in the top menu.
3. Click **Configure MCP Server**.



4. Enter the URL for your MCP server in the **Add a new MCP server** field.
5. Click **Test connection**.
6. If the connection is successful, you see a list of tools the server exposes. Scroll to the bottom of that list, to give the server a name and configure authentication or additional headers.
7. Click **Save** to persist the configuration and start using it in agents.

The system connects to the MCP server and discovers available tools.

INFO

Currently, only MCP servers that support SSE (Server-Sent Events) are compatible. HTTP-based MCP servers are planned for future releases. Only tools are supported, not prompts or resources.

IMPORTANT

Forwarding authentication headers to third-party MCP servers is a security risk. Enable this option only if the server is trustworthy.

CREATING CUSTOM MCP SERVERS

To extend the AI Agent Manager with custom tools, develop an MCP server that:

1. Exposes an SSE endpoint.
2. Implements the MCP protocol.
3. Defines tool schemas with clear descriptions.
4. Executes tool logic and returns structured results.

For MCP server implementation details, refer to the Model Context Protocol specification. You can also find an article on how to build MCP servers with NestJS and publish them on the Cumulocity platform as a microservice in the [Cumulocity TechCommunity](#).

BEST PRACTICES

- **Tool naming:** Use clear, descriptive names for custom MCP tools. The agent uses tool names and descriptions to decide when to call them.
- **Parameter validation:** Ensure MCP tools validate parameters and handle errors gracefully. Return clear error messages that help the agent understand what went wrong.

REST API

The AI Agent Manager provides a REST API for programmatic management of agents, providers, and conversations. This allows you to integrate AI agents into your own applications and workflows.

API OVERVIEW

The API is exposed by the AI Agent Manager microservice at:

```
/service/ai/
```

All API endpoints require authentication with valid Cumulocity credentials. The API supports standard CRUD operations for agents, tools, and providers that mostly require the `ROLE_AI_AGENT_ADMIN` permission. You best explore the full CRUD API via the OpenAPI documentation. You access this documentation in multiple ways:

- Download the OpenAPI JSON by accessing `/service/ai/api-json`. This file then loads into a tool like Postman or similar.
- Install the **AI Agent Manager OpenAPI** plugin from AI-Plugins in your **Administration > Ecosystem > Extensions** view into a cloned application. This lists all endpoints.

The following documentation only covers how to integrate agents that are already created into any UI or third-party server. All endpoints are explained in the OpenAPI documentation.

INTERACTING WITH AGENTS

To talk to an agent, you only require the `ROLE_AI_AGENT_READ` permission. The idea is that any end-user can talk to an agent, but not all users can create, update, or delete such agents. The following examples show how to use the API to talk to an already defined agent:

Simple prompt:

```
POST /service/ai/agent/text/{agent-name}
{
  "prompt": "What is the temperature of device 12345?"
}
```

Returns a plain text response.

With variables:

Use variables to inject dynamic data into prompts or system prompts:

```
POST /service/ai/agent/text/{agent-name}
{
  "variables": {
    "deviceId": "12345"
  },
  "prompt": "What is the status of device {{deviceId}}?"
}
```

Define variables in the system prompt using double curly brackets: `{{variableName}}` .

Maintaining conversations:

Use the `messages` array to maintain conversation context:

```
POST /service/ai/agent/text/{agent-name}
{
  "messages": [
    { "role": "user", "content": "Hi!" },
    { "role": "assistant", "content": "Hello! How can I help you?" },
    { "role": "user", "content": "Tell me about device 12345." }
  ]
}
```

The agent uses the conversation history to provide contextual responses.

Full response format:

Add `?fullResponse=true` to receive a detailed JSON response:

```
POST /service/ai/agent/text/{agent-name}?fullResponse=true
{
  "prompt": "Hello"
}
```

The full response includes:

- Text response
- Tool calls and results
- Token usage statistics
- Reasoning steps (if enabled)
- Provider metadata

INFO

The same APIs exist for an object agent. Only replace `text` with `object` in the URL.

STREAMING

By switching the content type to a streaming type, you get a streaming response from the API. Two modes are currently supported:

- Text only is returned if no query parameter is attached.
- A data SSE stream that also includes tool calls and status information is returned if the `?fullResponse=true` parameter is set.

Add the `text/event-stream` or `application/x-ndjson` accept header to force streaming. A `fullResponse` return is determined by two line breaks and the keyword `data: .`

As an example a call for `hello world` with `?fullResponse=true` returns:

```

data: {"type":"start"}

data: {"type":"start-step","request":{"body":{"model":"claude-sonnet-4-5","max_tokens":64000,"system":[{"type":"text","text":"A test"}],"messages":[{"role":"user","content":[{"type":"text","text":"hello world"}]}],"stream":true},"warnings":{}}

data: {"type":"text-start","id":"0"}

data: {"type":"text-delta","text":"Hello! ","id":"0"}

data: {"type":"text-delta","text":"","id":"0"}

data: {"type":"text-delta","text":"How ","id":"0"}

data: {"type":"text-delta","text":"can ","id":"0"}

data: {"type":"text-delta","text":"I ","id":"0"}

data: {"type":"text-delta","text":"help ","id":"0"}

data: {"type":"text-delta","text":"you ","id":"0"}

data: {"type":"text-delta","text":"today?","id":"0"}

data: {"type":"text-end","id":"0"}

data: {"type":"finish-step","finishReason":"stop","usage":
{"inputTokens":11,"outputTokens":16,"totalTokens":27,"cachedInputTokens":0},"providerMetadata":{"anthropic":{"usage":
{"input_tokens":11,"cache_creation_input_tokens":0,"cache_read_input_tokens":0,"cache_creation":
{"ephemeral_5m_input_tokens":0,"ephemeral_1h_input_tokens":0,"output_tokens":16,"service_tier":"standard"},"cacheCreationInputTokens":0,"stopSe
quence":null,"container":null}},"response":{"id":"msg_017QpgqQMUEbCs7MastWLP","timestamp":"2026-02-04T16:44:09.231Z","modelId":"claude-
sonnet-4-5-20250929","headers":{"anthropic-organization-id":"3f82821e-6dbf-463c-934f-a2f81b555764","anthropic-ratelimit-input-tokens-
limit":"2000000","anthropic-ratelimit-input-tokens-remaining":"2000000","anthropic-ratelimit-input-tokens-reset":"2026-02-04T16:44:08Z","anthropic-
ratelimit-output-tokens-limit":"400000","anthropic-ratelimit-output-tokens-remaining":"400000","anthropic-ratelimit-output-tokens-reset":"2026-02-
04T16:44:08Z","anthropic-ratelimit-tokens-limit":"2400000","anthropic-ratelimit-tokens-remaining":"2400000","anthropic-ratelimit-tokens-reset":"2026-02-
04T16:44:08Z","cache-control":"no-cache","cf-cache-status":"DYNAMIC","cf-ray":"9c8ba186efd17fa0-FRA","connection":"keep-alive","content-security-
policy":"default-src 'none'; frame-ancestors 'none';","content-type":"text/event-stream; charset=utf-8","date":"Wed, 04 Feb 2026 16:44:09 GMT","request-
id":"req_011CXoN8S7HVR6WWGFy7r32J","server":"cloudflare","strict-transport-security":"max-age=31536000; includeSubDomains; preload","transfer-
encoding":"chunked","x-envoy-upstream-service-time":"973","x-robots-tag":"none"}}}

data: {"type":"finish","finishReason":"stop","totalUsage":{"inputTokens":11,"outputTokens":16,"totalTokens":27,"cachedInputTokens":0}}

```

While a call without `?fullResponse=true` simply returns:

```
Hello!           How can I help you today?
```