

CodeArts Agent

User Guide (IDE)

Issue	01
Date	2026-07-29



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2026. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: <https://www.huaweicloud.com/intl/en-us/>

Contents

1 What Is CodeArts Agent IDE?.....

2 Quick Start.....

3 Agents.....

3.1 Built-in Agents.....

3.2 Custom Agents.....

3.3 Agent Chats.....

3.4 Sandbox.....

3.5 Multi-Chat Management.....

3.6 Checkpoints.....

3.7 Diff View.....

3.8 LSP.....

4 Models.....

5 Code Generation.....

6 Intelligent Q&A.....

7 Scheduled Tasks.....

8 Context.....

8.1 Adding Context.....

8.2 Rules.....

8.3 Knowledge Bases.....

8.4 Multimodal Input.....

8.5 Memory.....

8.6 Repository Index.....

9 MCP.....

10 Skills.....

11 Web-based Preview.....

12 Remote Development.....

12.1 SSH Development.....

12.2 WSL Distributions.....

12.3 Local Docker Container.....

1

2

6

6

9

24

36

45

46

51

52

54

60

62

71

74

74

78

92

107

110

112

119

127

146

150

150

156

159

13 Chat History..... 162

14 Hooks..... 166

15 Slash Commands..... 182

16 Intelligent Programming..... 190

16.1 Code Review..... 190

16.2 Code Fixing..... 194

17 Settings..... 199

17.1 IDE Tool Panel..... 199

17.2 Global Settings of the IDE.....203

17.3 Process Resources Explorer..... 215

17.4 Network Proxy.....217

18 Obtaining Logs..... 219

19 Feedback..... 224

1 What Is CodeArts Agent IDE?

CodeArts Agent IDE, an AI programming environment within the CodeArts series, deeply integrates AI into the development toolchain to deliver an intelligent, efficient programming experience.

Supported OSs

For optimal compatibility and performance, the following OSs are recommended:

- Windows OS: **Windows 11 (x64) is recommended.** For Windows 10 (x64), the version must be 2019 or later, and upgrading to the latest stable version is advised.
- macOS: macOS 11 or later, compatible with ARM64 (Apple Silicon).
- Linux: Linux (ARM64) architecture, glibc version ≥ 2.31 , glibcxx version $\geq 3.4.26$.

Dual Development Modes

CodeArts Agent offers two flexible development modes to meet different scenario requirements.

- **IDE mode:** follows traditional workflows, letting developers fully control the development pace and lifecycle.
- **Space mode (CodeArts Agent Space):** the built-in AI assistant mode in CodeArts Agent IDE that extends beyond coding to provide general AI support. Developers can write documents, analyze data, and design visually in an independent chat interface. For details, see [What Is CodeArts Agent Space?](#)

Security Assurance

CodeArts Agent IDE provides sandbox capabilities, offering dual security for AI-collaborative development.

- Data scope control: AI-accessible data is strictly isolated to prevent sensitive information leaks.
- Command risk control: Permissions for executing high-risk commands are restricted to mitigate system risks from misoperations or malicious code.

2 Quick Start

This section describes how to install CodeArts Agent IDE to quickly start intelligent coding.

Preparations

- Before coding with CodeArts Agent IDE, **apply for the open beta and purchase a package.**
 - a. Apply for the open beta. For details, see open beta application.
 - b. After the OBT application is approved, subscribe to a CodeArts Agent package.
CodeArts Agent is available in two editions: **basic** and **professional**. Select a proper edition based on your service requirements. For details about how to subscribe to a package, see subscription management.
- Install a browser, such as Chrome.
During IDE login, the system redirects you to a browser for account verification. To ensure a smooth process, deploy and configure an available browser in the Linux environment in advance.

Installing CodeArts Agent IDE

Step 1 Download the CodeArts Agent IDE installation package from the official website of CodeArts Agent.

Step 2 Double-click the package and complete the installation as prompted.

If you need to install the package applicable to the Linux (ARM64) architecture and the double-click installation fails, you can install it using the command line.

On the Linux terminal, navigate to the directory containing the installation package. Run the following command to perform installation:

```
sudo dpkg -i codearts-agent-linux-***.deb
```

Step 3 Log in to CodeArts Agent IDE and start intelligent coding.

If this is the first time you use CodeArts Agent IDE, the welcome page will be displayed. Complete the configuration as prompted.

1. On the welcome page, click **Get Started**. The import configuration page is displayed.

2. Select **Import from VS Code** or **Skip**. In this example, click **Skip**.
If you have created a project in Visual Studio Code, click **Import from VS Code** and import the project to CodeArts Agent IDE.
3. Click **Log In**. The Huawei Cloud login page is displayed.
4. Enter your account name and password and log in.
After a message is displayed indicating that the login is authorized and successful, return to CodeArts Agent IDE.

If this is not your first time to use CodeArts Agent IDE, log in by performing the following operations:

1. Click **Log In**. The Huawei Cloud login page is displayed.
2. Enter your account name and password and log in.
Wait until the login is successful and then return to CodeArts Agent IDE.

Step 4 (Optional) Log out of CodeArts Agent IDE.

To log out, click  in the upper right corner of CodeArts Agent IDE and choose **Log Out**.

----End

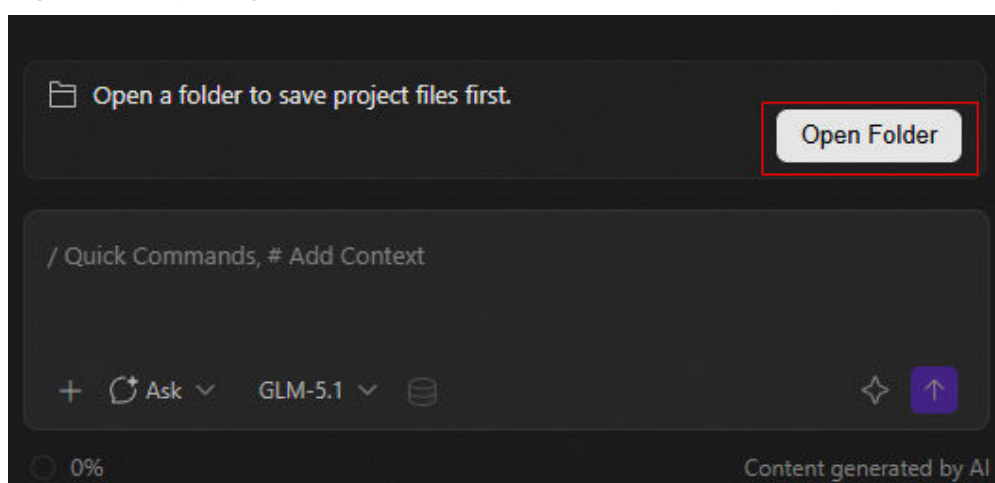
Creating or Opening a Project

After installing CodeArts Agent IDE, you can start your development in any of the following ways:

Importing a local folder: Import local project files to the IDE to quickly start development.

1. In the upper left corner of the IDE, choose **File > Open Project**.
You can also click **Open Folder** in the lower right corner on the chat panel of CodeArts Agent IDE.

Figure 2-1 Opening a folder



2. Select a local folder and open it in the IDE.

If you are using the IDE for the first time, you need to select **Yes, I trust the authors** in the displayed dialog box to trust the folder and enable all functions.

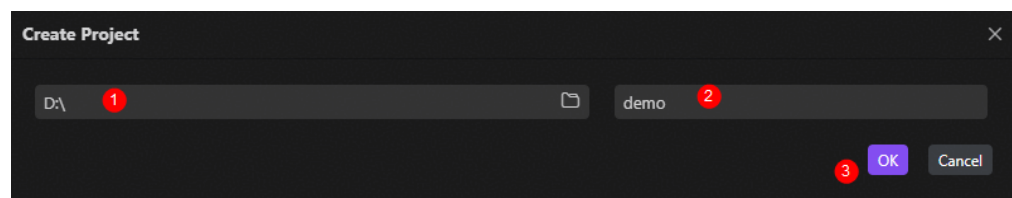
Creating a project: Create a project from scratch and customize your development environment.

1. In the upper left corner of the IDE, choose **File > New > Create Project**.
2. Select a path for storing the project, enter the project name, and click **OK**.

If you are using the IDE for the first time, you need to select **Yes, I trust the authors** in the displayed dialog box to trust the folder and enable all functions.

The project name must start with a letter, and can contain a maximum of 64 characters. Letters, digits, hyphens (-), and underscores (_) are allowed.

Figure 2-2 Creating a project

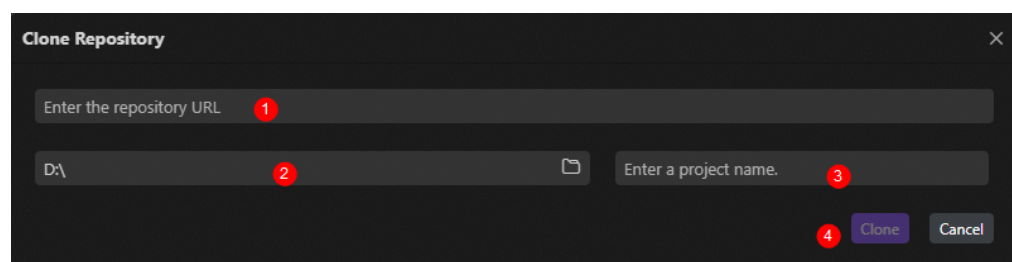


Cloning a Git repository: Clone a remote repository using Git to easily continue developing an existing project.

1. In the upper left corner of the IDE, choose **File > New > Clone Repository**.
2. Enter the repository URL, select the local storage location of the repository, and click **Clone**.

If the message "Could not connect to server" is displayed during the cloning process, resolve the issue as instructed in "Could not connect to server" Is Displayed When Cloning a Git Repository. Then perform the cloning again.

Figure 2-3 Cloning a repository



3. Enter the username and password of the to-be-cloned repository as prompted and click **Continue** to clone it to the local host and open it in the IDE.

Related Operations

- If Microsoft Store briefly flashes and does not respond when you double-click the .exe installer of the IDE, see [What Do I Do If the .exe Installer Crashes After Being Double-Clicked in Windows?](#)

- To uninstall CodeArts Agent IDE installed in Linux, see [How Do I Install CodeArts Agent IDE on Linux Using Commands?](#)
- For how to convert Windows line breaks (CRLF) to Linux-recognizable line breaks (LF), refer to [How Do I Handle Windows CRLF Line Endings That Cause Execution Errors in Linux?](#)
- CodeArts Agent displays message "Restricted mode. Functions not available." Handle it by referring to [How Do I Handle the Error "Restricted mode. Functions not available."?](#)

3 Agents

3.1 Built-in Agents

Agents are programming assistants tailored for diverse development scenarios. CodeArts Agent provides multiple out-of-the-box agents that require no manual creation, aiming to provide efficient programming assistance for various development scenarios.

Built-in Primary Agents

Table 3-1 Built-in primary agents

Name	Creator	Description
Agent	System built-in	The core functional module of CodeArts Agent, supporting the understanding of complex developer requirements and autonomous execution path planning. It applies to end-to-end development scenarios (such as requirement analysis → code generation → test verification), achieving full-process automation to significantly boost R&D efficiency.
AgentTeam	System built-in	A multi-agent collaboration engine featuring the architecture of "Leader intelligent orchestration + Teammate autonomous execution." It decomposes complex tasks into parallel subtasks, allowing multiple specialized agents to collaborate like a real team. The entire process is event-driven and visualized, without manual intervention. AgentTeam can only be used in CodeArts Agent Space. For details, see Multi-task Parallelism.

Name	Creator	Description
Intelligent Q&A	System built-in	An interactive knowledge assistant based on large language models, supporting questions in natural languages and accurate answers.

Built-in Sub-agents

In CodeArts Agent, a sub-agent acts as an AI assistant to the primary agent, designed to handle specific types of tasks and return the results to the primary agent. Sub-agents are primarily used in the following scenarios:

- **Long-chain complex tasks:** Sub-agents are applicable to lengthy processes and complex logic, where they autonomously break down the task, plan steps, and implement them, such as full-chain code development.
- **Context Isolation:** Sub-agents isolate sub-task contexts to prevent the primary agent from context redundancy, token bloating, and context overflow, such as in batch data parsing.
- **Vertical domain expertise:** Each sub-agent can be configured with exclusive role prompts and private knowledge bases to develop domain expertise, enhancing inference precision and task accuracy, such as a security audit expert sub-agent.
- **Permission isolation:** Sub-agents are configured with independent access permissions to achieve isolation for sensitive data, private knowledge bases, high- and low-risk interfaces, file resources, and project data, ensuring data and system security, such as permission isolation for high-risk operations like data deletion and system commands.

Table 3-2 Built-in sub-agents

Name	Description
general	A general-purpose agent used to research complex problems and execute multi-step tasks.
explore	A code repository exploration agent dedicated to quickly locating files, searching code keywords, and answering related questions.
spec-task-agent	An agent used to generate implementation tasks based on requirements and design.
spec-requirement-agent	An agent used to generate requirements for the Easy Approach to Requirements Syntax (EARS) format based on the project description and context.
spec-design-agent	An agent used to generate comprehensive technical designs, converting requirements (what to do) into architectures (how to do it).

Name	Description
developer-test-agent	An agent dedicated to unit test generation, repair, coverage optimization, and review. Prompt-based calling: Enter a command directly into the chat box to invoke it. For example, use developer-test-agent to generate a unit test for a <i>specific</i> method in a <i>specific</i> file.
bug-fix-agent	An agent used to parse software issues from user bug reports and related artifacts, focusing on issue analysis, fault localization, and patch implementation.
rule-generator	Rule generator.

Modifying the Basic Model of a System Built-in Agent

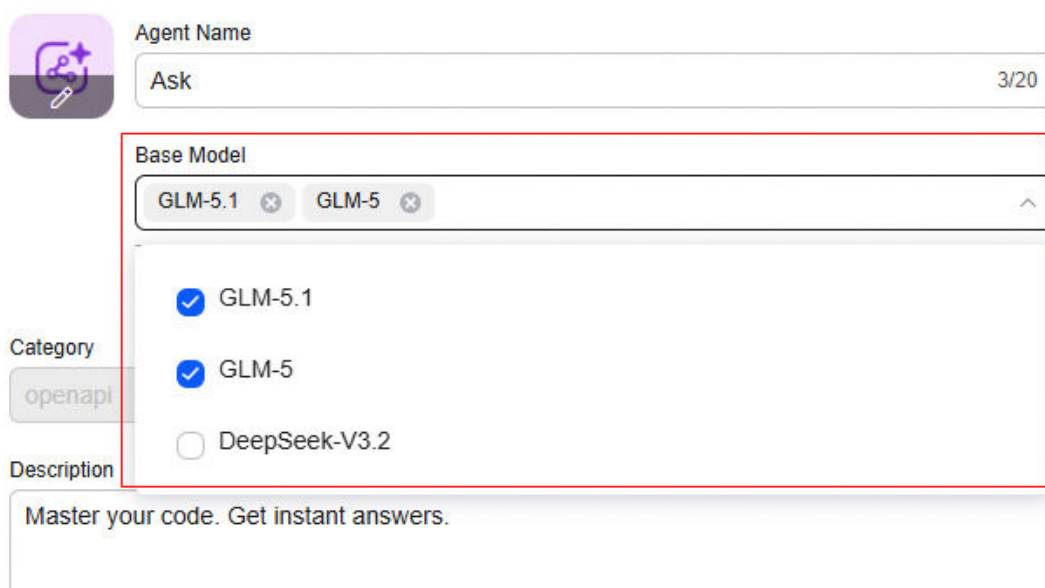
Only **enterprise administrators** of the CodeArts Agent **basic** or **professional** edition can modify the basic model of a system built-in agent.

On the **Agents** tab of the CodeArts Agent console, you can view details about the current agent and flexibly adjust the underlying basic model based on actual service requirements to achieve optimal performance.

- Step 1** Log in to the [CodeArts Agent console](#).
- Step 2** In the navigation pane on the left, choose **Agent Settings > Agents**.
- Step 3** In the agent list, click **Edit** next to a built-in agent. The model editing page is displayed.
- Step 4** In the basic model drop-down list, select the basic model of the agent and click **OK**.

Figure 3-1 Modifying a basic model

Application Information



Agent Name: Ask 3/20

Base Model

GLM-5.1 x GLM-5 x ^

☒ GLM-5.1

☒ GLM-5

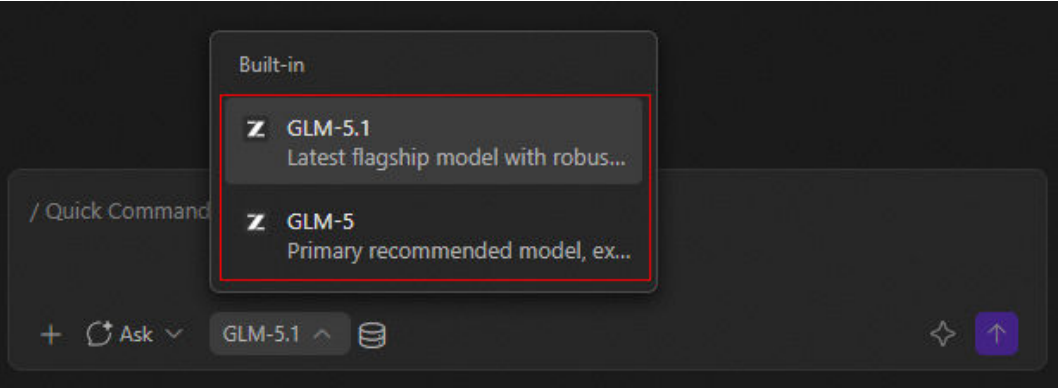
☐ DeepSeek-V3.2

Category: openapi

Description: Master your code. Get instant answers.

Step 5 Return to the chat panel of CodeArts Agent IDE and select the corresponding built-in agent. Verify that the model has been modified.

Figure 3-2 Modified basic models of the agent



----End

3.2 Custom Agents

Agents are programming assistants tailored for diverse development scenarios. CodeArts Agent not only provides various built-in agents but also supports custom agent capabilities. By configuring prompts, skills, and tool sets within a custom agent as required, you can create a dedicated assistant to handle complex development tasks more efficiently.

Classification of Custom Agents

Custom agents are classified into **local** and **cloud** agents by creation method, and are divided into **enterprise-level**, **team-level**, **project-level**, and **user-level** agents by scope of application. Select the most appropriate agent type to create based on your specific development scenarios and collaboration needs.

Table 3-3 Classification of custom agents

Creat ion Meth od	Scop e of Appli catio n	Description
Local	Projec t	Effective only for the current project, distributed along with the repository, and stored locally. Storage directory: <code>./ .codeartsdoer/agents</code> in the root directory of the project
	User	Effective for all projects of the current user, and stored locally Local storage directory: <code>%USERPROFILE%\ .codeartsdoer/agents/</code>

Creation Method	Scope of Application	Description
Cloud	Enterprise	Available to all enterprise users, created by enterprise administrators, and stored in the cloud. When a user logs in to the CodeArts Agent client, the system automatically caches the cloud agents accessible to the user to the local directory: %USERPROFILE%/.codeartsdoer/cache/enterprise/agents/ .
	Team	Restricted to designated teams, created by enterprise or team administrators, and stored in the cloud. When a user logs in to the CodeArts Agent client, the system automatically caches the cloud agents accessible to the user to the local directory: %USERPROFILE%/.codeartsdoer/cache/team/agents/ .
	User	Available only to the creator and stored in the cloud. When a user logs in to the CodeArts Agent client, the system automatically caches the cloud agents accessible to the user to the local directory: %USERPROFILE%/.codeartsdoer/cache/user/agents/ .

Creating Custom Agents on the GUI

CodeArts Agent supports creating both **local** and **cloud** agents. When a user logs in to the CodeArts Agent client, the system caches the cloud agents accessible to the user to the local. After that, the local cache automatically refreshes **every 24 hours** to ensure synchronization with the cloud. You can also manually click **Refresh** on the settings page of the CodeArts Agent client to refresh the accessible cloud agents in the local cache.

Creating a Local Custom Agent

- Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
- Step 3** In the navigation pane on the left, choose **Agents**.
- Step 4** Click **Create Agent**.
- Step 5** Set agent parameters and click **Create**.

Figure 3-3 Creating a local custom agent

Agents / Create Agent

*Agent Name

Enter an agent name.

Scope

User

Agent Type

Primary/Sub-agent

*Description

Describe what the agent can do.

0/5000

Prompt

(Optional) Enter the agent's role, tone, workflow, tool preferences, and rules/specifications.

0/10000

Skills

+ Associate Skills

No skills associated. Add one to execute specialized tasks.

Tools

☒ Built-in tools

☒ Read

Retrieve and view files

☒ Edit

Add, delete and edit files

Table 3-4 Parameters for creating a local custom agent

Parameter	Description
Agent icon	The system provides various preset icons for agents. Click on the icon to select a preferred style.
Agent name	Name of the custom agent to be created. Naming convention: 2 to 20 characters, consisting of letters, digits, underscores, and hyphens (-), and must start with a letter.

Parameter	Description
Scope	Application scope of the custom agent. • Project: The agent is only available within the current project and cannot be called by other projects. • User: The agent can be called across all projects of the current user.
Agent type	Working mode of the agent, which affects its behavior logic and capability boundary. • Primary/Sub-agent: It can be used as a sub-agent to be called by a primary agent or as a primary agent to call a sub-agent. • Primary agent: It runs independently to process user requests and cannot be called by other agents. • Sub-agent: It can be called by primary agents to perform specific tasks.
Description	A brief explanation of what the agent does to help users understand its functionality, with a maximum of 5,000 characters.
Prompt	System instructions that define the role, tone, workflow, and rules/specifications of the agent.
Skills	Skills associated with the agent to enable it to perform certain specialized tasks. NOTE CodeArts Agent has no hard limit on the number of associated skills, but it is recommended that the number not exceed 20 to ensure optimal user experience.
MCP tools	This configuration item is displayed only when Scope is set to Project. External tools or services connected to the agent to enable it to interact with the outside world, such as obtaining real-time information and operating software. Up to 80 MCP tools can be added. It is recommended that no more than eight MCP tools be enabled at the same time, with three being ideal for the optimal user experience.

Parameter	Description
Built-in tools	Built-in, out-of-the-box tools provided by the agent. • Read: used to retrieve and view files. • Edit: used to add, modify, or delete files. • Terminal: used to run system commands in the command line environment. • Preview: used to preview the generated frontend page in real time. • Webfetch: used to capture and read web page content.

Step 6 Verify the custom agent.

- After the custom agent is created, you can view it on the corresponding category tab (such as **Project** or **User**), and its status will be displayed as enabled.
- In the input box on the chat page, click **Agent**. Verify that the target agent is displayed in the custom agent list.

 NOTE

If you create a **sub-agent**, it will not be displayed in the custom agent list.

-----End

Creating a Cloud Custom Agent

Step 1 Log in to the [CodeArts Agent console](#).

Step 2 In the navigation pane on the left, choose **Agent Settings > Agents**.

Step 3 Click **Create Agent**.

Step 4 Set agent parameters and click **Create and Enable**.

Table 3-5 Parameters for creating an agent

Parameter	Description
Agent icon	The system provides various preset icons for agents. Click  on the icon to select a preferred style.
Agent name	Name of the custom agent to be created. Naming convention: 2 to 20 characters, consisting of letters, digits, underscores, and hyphens (-), and must start with a letter.

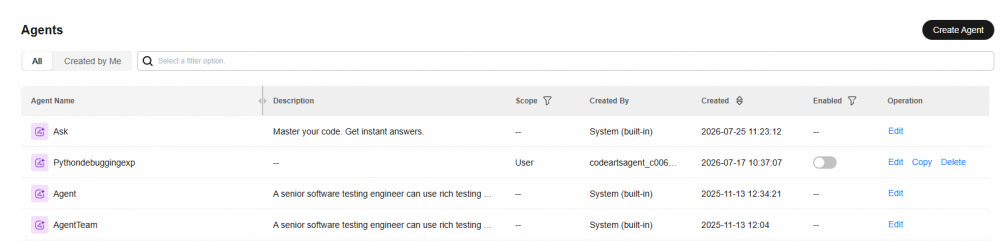
Parameter	Description
Agent type	Working mode of the agent, which affects its behavior logic and capability boundary. • Primary/Sub-agent: It can be used as a sub-agent to be called by a primary agent or as a primary agent to call a sub-agent. • Primary agent: It runs independently to process user requests and cannot be called by other agents. • Sub-agent: It can be called by primary agents to perform specific tasks.
Description	A brief explanation of what the agent does to help users understand its functionality, with a maximum of 5,000 characters.
Applied to	Application scope of the custom agent, which cannot be modified after creation. • Individual: Only the creator can use it. • Team: It is available only to specified teams and can be created by enterprise administrators or team administrators. • Enterprise: It is available to all enterprise members and can be created only by enterprise administrators.
Available team space	This parameter is displayed only when Applied To is set to Team. It specifies the teams that can use the agent. Enterprise administrators can view all teams but can select only one, while team administrators can only view and select the team they belong to.
Prompt	System instructions that define the role, tone, workflow, and rules/specifications of the agent.
Built-in tools	Built-in, out-of-the-box tools provided by the agent. • Read: used to retrieve and view files. • Edit: used to add, modify, or delete files. • Terminal: used to run system commands in the command line environment. • Preview: used to preview the generated frontend page in real time. • Webfetch: used to capture and read web page content.

Parameter	Description
Skills	Skills associated with the agent to enable it to perform certain specialized tasks. NOTE CodeArts Agent has no hard limit on the number of associated skills, but it is recommended that the number not exceed 20 to ensure optimal user experience.

Step 5 View the created agent.

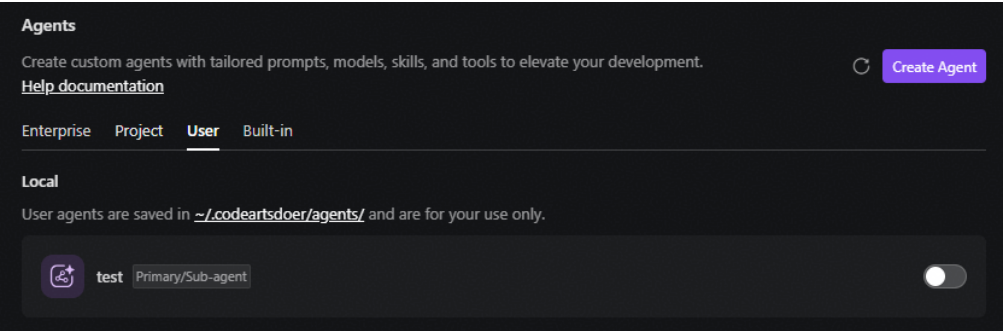
- Return to the **Agent Management** page. You can view the created agent in the agent list.

Figure 3-4 Viewing the target agent on the Agents page



- On CodeArts IDE, open the settings page and click **Agents**. Then click  to manually refresh the page. After the refresh is complete, open the target tab (for example, **User**) to view the target agent.

Figure 3-5 Viewing the target agent on the settings page



- In the input box on the chat page, click **Agent**. Verify that the target agent is displayed in the custom agent list.

 NOTE

If you create a **sub-agent**, it will not be displayed in the custom agent list.

----End

Creating Custom Agents Using Markdown Files

CodeArts Agent allows you to create **local project or user agents** through Markdown files. **When project and user agents share the same name, the project agent takes precedence over the user agent.**

Agent File Format

Agents are written in the Markdown file format, with YAML Frontmatter metadata configured at the top of the file and with prompt instructions contained in the body.

```
---
name: agent-name      # Agent name
mode: subagent        # Agent mode
description: Agent desc # Agent function description
tools:                # Tool permission configuration
  tool1: true          # Allows the use of tool1.
  tool2: false         # Disables the use of tool2.
---
```

```
# Prompt
Here lists the system prompt content of the agent.
```

The following table describes how to set the agent parameters.

Table 3-6 Markdown file parameters

Parameter	Mandatory (Yes/No)	Parameter Type	Default Value	Description
name	Yes	String	N/A	Name of the agent to be created.
mode	No	String	N/A	Role of the agent. - subagent: The agent can only be used as a sub-agent and can be called by a primary agent. - primary: The agent can only be used as a primary agent and cannot be called by other agents. - all: The agent can be used as a sub-agent to be called by a primary agent or as a primary agent to call a sub-agent.
description	Yes	String	N/A	Description of the agent.
tools	No	Object	N/A	Permissions for tools that can be used by the agent. Table 3-8 lists the tools supported by CodeArts Agent. You can also use * to match all tools. Example: tools: "":false #All tools are disabled.
Prompt	Yes	String	-	System prompt of the agent.

Table 3-7 Value description for tools

Parameter	Description
true	Allowed to use a tool.
false	Not allowed to use a tool.

Table 3-8 Tools supported by CodeArts Agent

Tool	Description
read	Reading files
write	Writing data to files
edit	Editing files
bash	Running commands
glob	File pattern matching
grep	Retrieving content
list	Listing directories
task	Sub-task
webfetch	Obtaining webpage content
websearch	Searching webpages
question	Inquiring users

Adding Project Agents

An agent customized for the current project is distributed along with the repository. **A project agent takes effect immediately after being added. If it is a sub-agent, when it is used is determined by the primary agent.** You can add a project agent using either of the following methods:

- **Method 1: Creating a Project Agent**
 - a. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
 - b. In the `./codeartsdoer/agents` folder of the explorer, right-click and choose **New > File** from the shortcut menu to create a Markdown file.
The agent name consists of lowercase letters, digits, and hyphens (-). It cannot start or end with a hyphen, cannot contain consecutive hyphens, and must be 1 to 64 characters in length. For example, code-reviewer.
 - c. Write the agent in the `code-reviewer.md` file and press the shortcut keys (Windows/Linux: Ctrl+S; macOS: Command(⌘)+S) to save the file.
- **Method 2: Placing an Existing Agent to the Specified Directory**

Place an existing agent to `./codeartsdoer/agents` of the project root directory.

Adding User Agents

A user agent takes effect immediately after being added. If it is a sub-agent, when it is used is determined by the primary agent. You can add a user agent using either of the following methods:

- **Method 1: Creating a User Agent**

- a. Create a Markdown agent file in the local directory `%USERPROFILE%/.codeartsdoer/agents/`.

The agent name consists of lowercase letters, digits, and hyphens (-). It cannot start or end with a hyphen, cannot contain consecutive hyphens, and must be 1 to 64 characters in length. For example, code-analyzer.

- b. Write the agent in the `code-analyzer.md` file and press the shortcut keys (Windows/Linux: Ctrl+S; macOS: Command(⌘)+S) to save the file.

- **Method 2: Placing an Existing Agent to the Specified Directory**

Place an existing agent to `%USERPROFILE%/.codeartsdoer/agents/`.

Example of Creating an Agent

The following describes how to create and use a **project sub-agent**.

Step 1 In the `./codeartsdoer/agents` folder of the file explorer, right-click and choose **New > File** from the shortcut menu to create a Markdown file named `code-analyzer.md`.

Step 2 Add the following content to `code-analyzer.md` and save the file.



CAUTION

The .md file name must match the name field within the file, otherwise your sub-agent will fail to run. For example, both the file name and the name field in the example are **code-analyzer**.

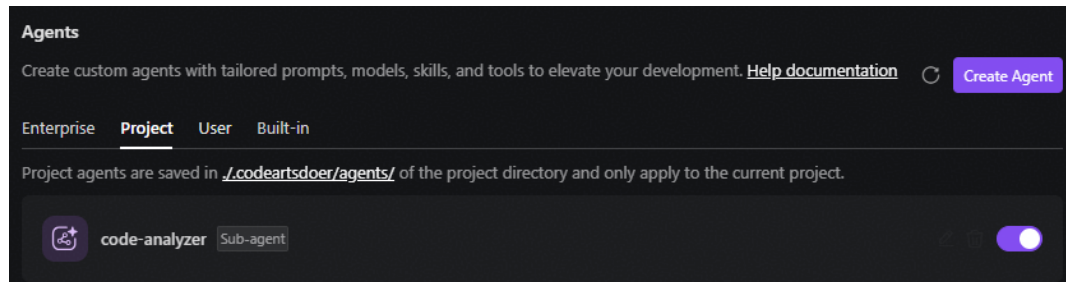
```
---
name: code-analyzer
mode: subagent
description: Code analysis expert for reading but not modifying
tools:
  read: true
  glob: true
  grep: true
  list: true
---
```

You are a code analysis expert. Your tasks are to:

1. Analyze code structure and quality.
2. Identify potential issues and areas for improvement.
3. Provide optimization suggestions.

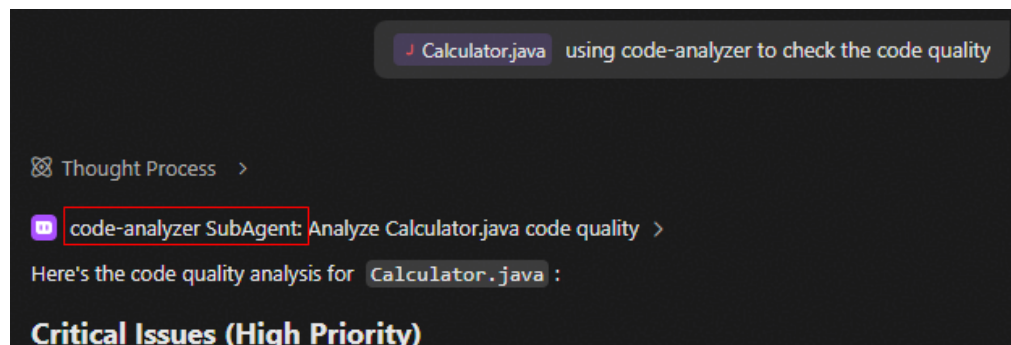
Analyze the code carefully, but do not modify any files.

After the file is saved, you can choose **Settings > Agents > Project** and find the target agent in the enabled state.

Figure 3-6 Viewing the target agent

Step 3 Add the code file to be checked and call a sub-agent to check the code quality.

1. On the chat panel of CodeArts Agent, enter #, click **File**, and select a file from the list.
2. Enter a prompt, for example, "use code-analyzer to check the code quality," and click .
3. View the sub-agent calling result.
The agent calls sub-agents and displays their names in the response.

Figure 3-7 Example of using the sub-agent

----End

Using Custom Agents

Step 1 At the bottom of the input box on the chat panel, select a custom agent (for example, a Python debugging expert) to switch to the custom agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

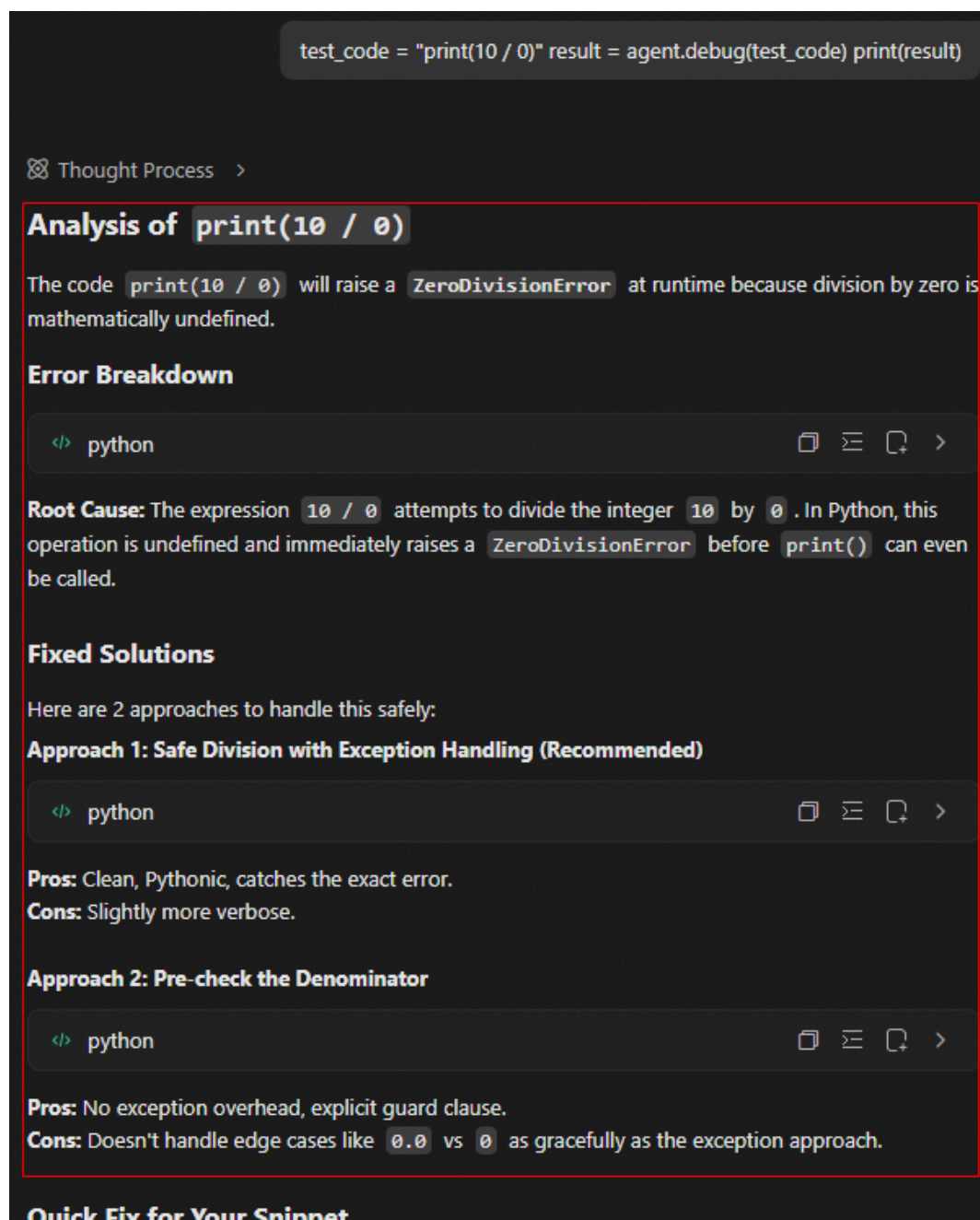
NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

Step 2 In the input box on the chat panel, enter a prompt (for example, a Python code block containing errors) and click  or press **Enter**.

Step 3 View the agent execution result.

The agent automatically detects syntax or logic errors in the code block and provides accurate correction suggestions and the corrected code.

Figure 3-8 Viewing the agent execution result

----End

Setting the Default Model for Custom Agents

Once a custom agent is created, its model usage adheres to the following rules: The **primary agent** defaults to the model used by the system's **built-in agent**. For example, if the built-in agent selects GLM-5 as its large language model, the custom agent will also use GLM-5. A **sub-agent** directly inherits the model used by the primary agent that invokes it. If you need to modify the default model used by a custom agent, you can do so by modifying the configuration file of the custom agent.

Step 1 Go to the directory where the custom agent's configuration file is stored.

For example, to modify a project custom agent, go to the `./codeartsdoer/agents/` directory of the project on the local. You can also locate the corresponding directory in the explorer of CodeArts Agent IDE. For details about the directories for storing Markdown files of other types of custom agents, see [Table 3-3](#).

Step 2 Open the Markdown file of the custom agent, add the model field, and save the file (Windows/Linux: Ctrl+S; macOS: Command(⌘)+S).

In CodeArts Agent, you can specify the default model for custom agents by configuring the model field. You can dynamically switch to the optimal model for specific tasks to balance performance and cost. For instance, employ a lightweight, high-speed model during the planning phase to boost response speed, and invoke a high-capability model during the implementation phase to ensure execution quality.

The value of the model field is in the format of *Provider/Model ID*, where *Provider* indicates the model provider (set to **inferhub-provider** in CodeArts Agent), and *Model ID* indicates the ID of the specific model.

 CAUTION

If the configured model does not exist or you do not have the permission, the **primary agent** automatically switches to the default model of the built-in agent, and the **sub-agent** inherits the model of the primary agent.

```
---
description: focusing on Python code error check and optimization, expected to be professional and concise.
mode: primary
model: inferhub-provider/GLM-5.1
permission:
  skill:
    '*': deny
disable: false
scope: project
avatar: avatar1
---
```

Table 3-9 IDs of built-in models in CodeArts Agent

Model Name	Model ID
GLM-5.1	GLM-5.1
GLM-5.2	GLM-5.2

Step 3 Return to the settings page of CodeArts Agent IDE. On the **Agents** tab, click  to refresh the agent configuration.

Step 4 View the default model of the custom agent.

In the input box on the chat panel, switch to the target custom agent. The system automatically applies the default model (for example, GLM-5.1) defined by the Markdown configuration file.

----End

Managing Custom Agents

CodeArts Agent IDE features full-lifecycle management capabilities for custom agents. You can efficiently control and continuously optimize created agents based on actual business needs.

Managing Local Custom Agents

Table 3-10 Managing local custom agents

Operation	Procedure
Enabling or disabling a local custom agent	On the settings page, choose Agents > Project or Agents > User . Click  or  to enable or disable an agent.
Editing a local custom agent	On the settings page, choose Agents > Project or Agents > User . Hover the cursor on the target agent and click  to edit the agent.
Deleting a local custom agent	On the settings page, choose Agents > Project or Agents > User . Hover the cursor on the target agent and click  . In the displayed dialog box, click OK . Deleted agents cannot be restored. Ensure that the agent is no longer needed before performing this operation.
Synchronizing a Custom Agent from the Cloud to the Local	If you log in to the CodeArts Agent client, the system will cache the cloud agents accessible to you locally. Then, the client will update the cache every 24 hours to keep the agents in sync with the cloud. You can also manually refresh them by clicking  on the Agents tab.
Enabling or disabling a custom agent synchronized from the cloud to the local	On the settings page, choose Agents > Enterprise or Agents > Team . Click  or  to enable or disable an agent.
Viewing the configuration of a custom agent synchronized from the cloud to the local	On the settings page, choose Agents > Enterprise , Agents > Team , or Agents > User > Cloud . Hover the cursor on the target agent, and click  to view the agent configuration details.

Managing Cloud Custom Agents

Table 3-11 shows the operation permissions of different roles on **managing cloud custom agents**.

Table 3-11 Operation permissions of different roles on cloud custom agents

Role	Enterprise Agent	Team Agent	User Agent (Operations Except Editing)	User Agent (Editing)
Enterprise administrators	All operations on all agents.	All operations on all agents.	All operations except editing on all agents.	Editing on agents created by themselves.
Team administrators	No operation permissions.	All operations on agents created by members of their own team.	All operations except editing on agents created by members of their own team.	Editing on agents created by themselves.
Members (of an enterprise or a team)	No operation permissions.	No operation permissions.	All operations except editing on agents created by themselves.	Editing on agents created by themselves.

Table 3-12 Managing cloud custom agents

Operation	Procedure
Enabling or disabling a cloud custom agent	1. Log in to the CodeArts Agent console. 2. In the navigation pane on the left, choose Agent Settings > Agents. 3. In the agent list, click  /  next to an agent to enable or disable it.
Editing a cloud custom agent	1. Log in to the CodeArts Agent console. 2. In the navigation pane on the left, choose Agent Settings > Agents. 3. In the agent list, click Edit in the Operation column of the target agent to edit it.

Operation	Procedure
Copying a cloud custom agent	When you need to reuse mature configurations, iterate multiple versions in parallel, share team templates, or isolate tasks, you can achieve this by copying agents. 1. Log in to the CodeArts Agent console. 2. In the navigation pane on the left, choose Agent Settings > Agents. 3. In the agent list, click Copy in the Operation column of the target agent. System built-in agents cannot be copied. 4. Adjust the configuration as required (or retain the default settings) and click Create and Enable to quickly generate a new agent.
Deleting a cloud custom agent	1. Log in to the CodeArts Agent console. 2. In the navigation pane on the left, choose Agent Settings > Agents. 3. Check the status of the target agent. • If it is enabled, disable it first. • If it is disabled, go to the next step to delete the agent. 4. Click Delete in the Operation column of the target agent. The Delete Agent dialog box is displayed. 5. Enter DELETE or click Autofill, and then click OK to delete the agent. Deleted agents cannot be restored. Ensure that the agent is no longer needed before performing this operation.

3.3 Agent Chats

CodeArts Agent's agent mode understands developer needs and plans tasks automatically. It enables full process automation from requirement analysis to coding, boosting efficiency in end-to-end development.

Core Capabilities

The agent offers essential features for understanding requirements, planning tasks, writing code, and using tools.

1. **Complex task planning and management:** Natural language can turn into well-structured action plans. The tool updates task lists instantly as work progresses, ensuring organized tracking and standardized handling of project activities.
2. **Autonomous coding:** The tool autonomously generates and modifies code, ensuring consistent updates across multiple files even in complex situations.

3. **Intelligent tool calling:** The agent retrieves information and performs tasks automatically. It accesses tools like code search, file reading, and shell commands to ensure the new code integrates smoothly with the project's current logic and dependencies.
4. **Interaction mechanism:** The tool includes a manual confirmation step at critical stages to maintain security and control during operations.

Core Development Modes

The agent provides the following development modes, which you can flexibly select based on your needs.

- **Vibe coding:** Coding while chatting, verify instantly. It is suitable for brainstorming and creative development, enabling rapid iteration within chats and dynamically capturing and clarifying potential requirements. It boosts efficient conversion from vague ideas into feasible prototypes.
- **Spec-driven development (SDD):** An engineering methodology that guides system design, development, and verification based on explicit technical or business specifications. Under this mode, all development activities follow established specifications to ensure that the final deliverable is highly consistent with the specifications in terms of functionality, performance, compatibility, and other aspects. It is suitable for scenarios with high requirements for consistency and verifiability.

Generating Code in Vibe Coding Mode

Step 1 Switch to the agent mode.

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

3. On the chat panel, click **Vibe Coding**.

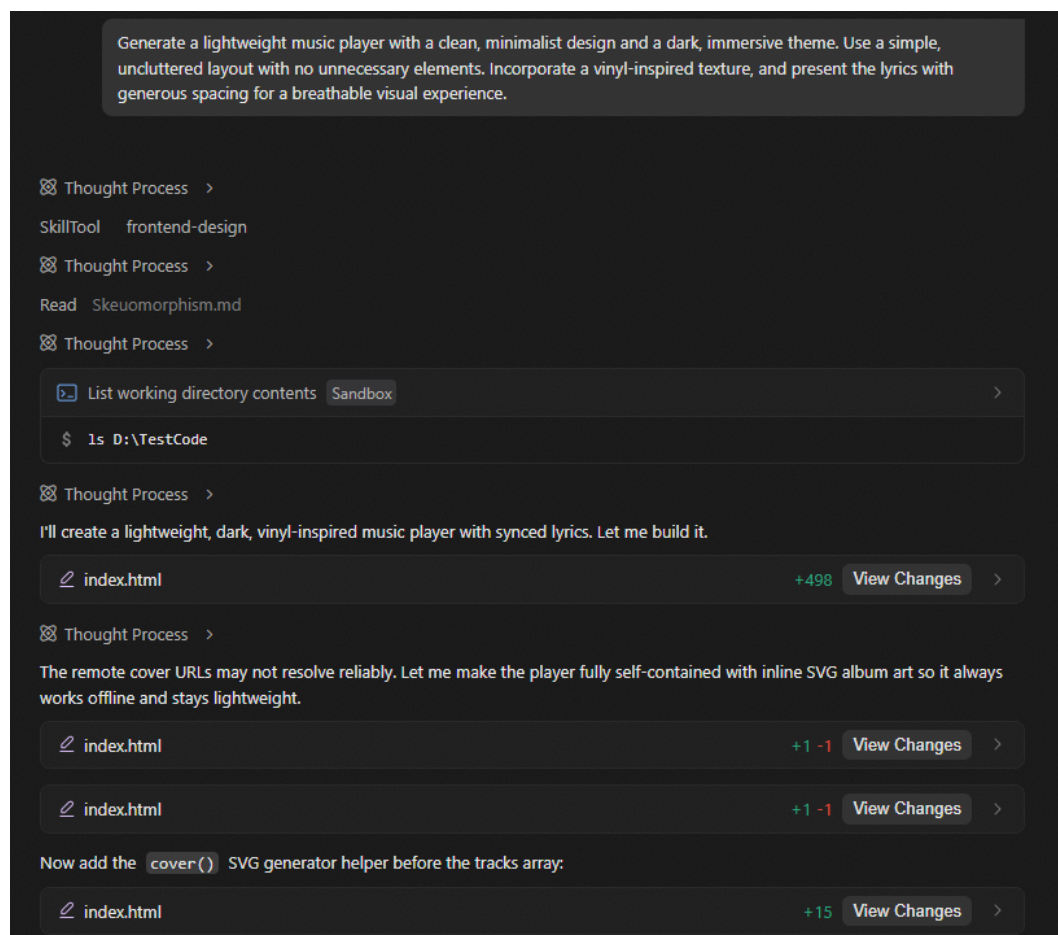
Step 2 Explain what you need. Type your task into the box, and click or press **Enter** to submit your request.

Generate a lightweight music player with a clean, minimalist design and a dark, immersive theme. Use a simple, uncluttered layout with no unnecessary elements. Incorporate a vinyl-inspired texture, and present the lyrics with generous spacing for a breathable visual experience.

NOTE

- During the response generation, you can initiate a new chat in the input box, and the new chat will be automatically added to the task queue. The task queue supports a maximum of **20 messages** waiting to be sent, and any exceeding messages will no longer be accepted.
- Different models may vary in their generated results; refer to the actual output for the final performance.

Step 3 Check and confirm the plan. The agent generates a to-do list based on your prompt.

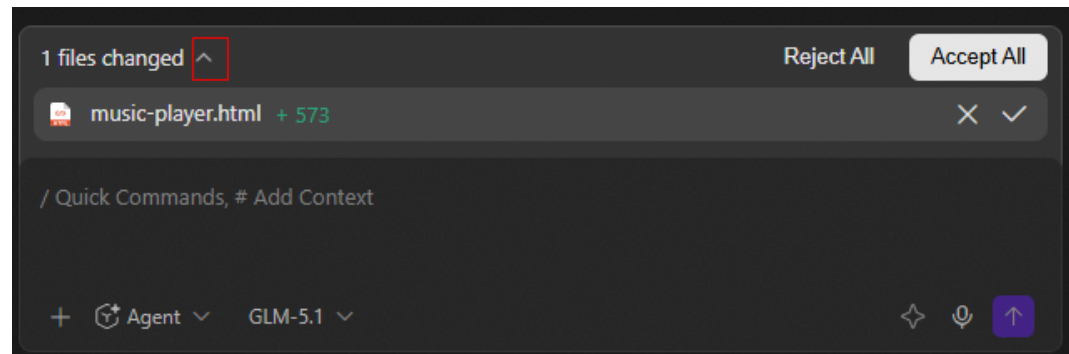
Figure 3-9 Execution plan

Step 4 Watch the process run. Once the plan is confirmed, the task begins step by step. The agent will walk you through each operation for confirmation. You can also expand the to-do list to view the progress of the current task.

- : The task is completed.
- : The task is in progress.
- : The task is not started.

Step 5 Confirm that tasks are completed. After the task is complete, the changed files are stored in the temporary storage area.

Step 6 Confirm the result. Click the arrow next to the file changed to expand the file information. Perform operations as required.

Figure 3-10 Result confirmation

- Click the file name to view the generated content.
- Move the cursor to the file, and click  to accept it or  to reject it.
- If there are multiple files, you can click the accept all button to accept all files or click the reject all button to reject all files.

----End

Generating Code in Spec-Driven Mode

Step 1 Switch to the agent mode.

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

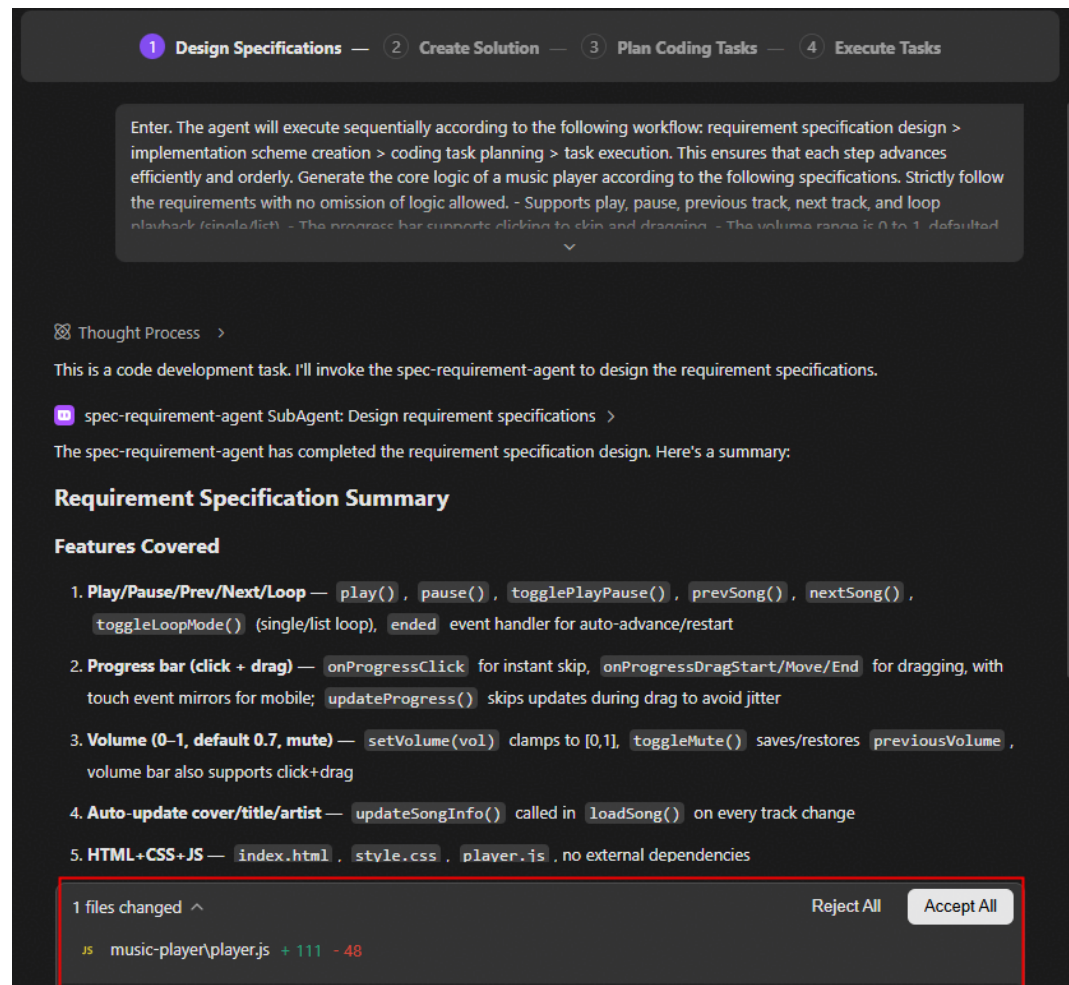
If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

3. On the chat panel, click **Spec-driven**.

Step 2 View the operations step by step following the preset workflow. In the input box, enter the task you want to complete and click or press **Enter**. The agent will execute sequentially according to the following workflow: requirement specification design > implementation scheme creation > coding task planning > task execution. This ensures that each step advances efficiently and orderly.

Generate the core logic of a music player according to the following specifications. Strictly follow the requirements with no omission of logic allowed.

- Supports play, pause, previous track, next track, and loop playback (single/list).
- The progress bar supports clicking to skip and dragging.
- The volume range is 0 to 1, defaulted to 0.7 and with mute toggling.
- Automatically updates the cover, song title, and artist information when switching songs.
- Implemented using HTML+CSS+JS

Figure 3-11 Executing a task based on a specified workflow

- If you confirm that the content of the requirements document basically meets your expectations, click **Accept All** or click  to accept the generated requirements document.
- If you do not approve the currently generated requirements document, click **Reject All** or click  to reject the generated requirements document. In this case, if the model cannot find the generated document in subsequent steps, it will generate the missing document again.
- After confirming that the requirements document is correct, click **Start Create Solution** to initiate the solution generation process.

----End

Generating Code Using Slash Commands

Beyond spec-driven development, CodeArts Agent can also use slash commands for code development.

Step 1 Switch to the agent mode.

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

- At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 NOTE

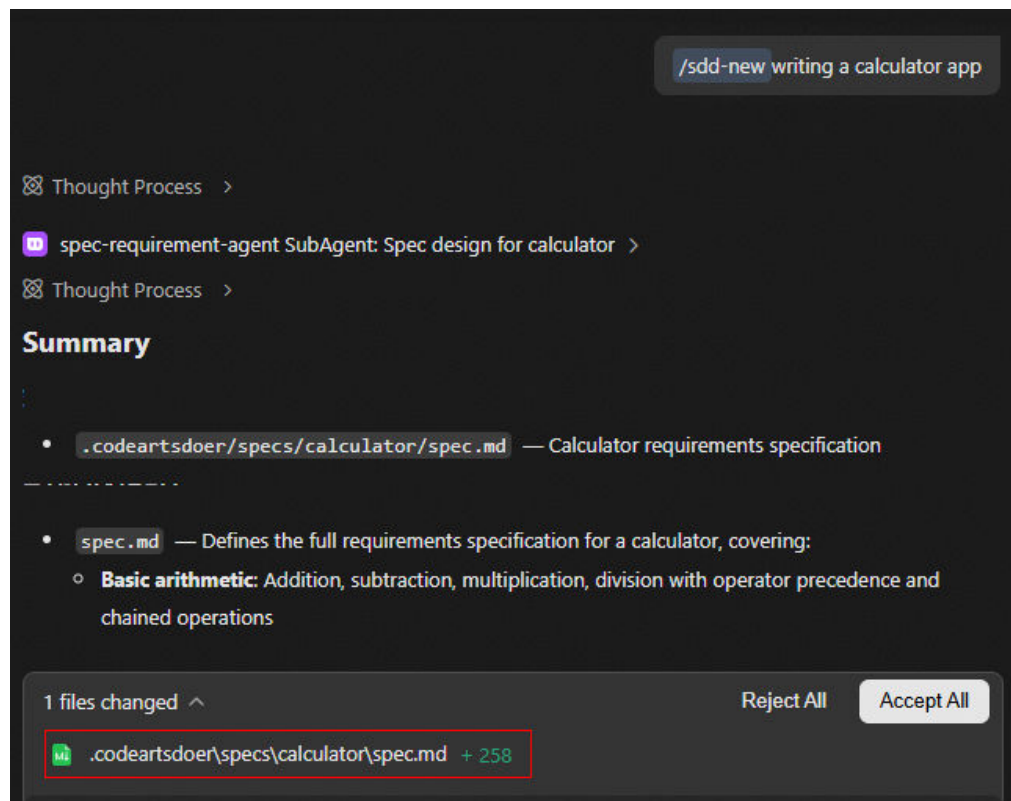
If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

- On the chat panel, click **Vibe Coding**.

Step 2 Create requirement specifications.

- Enter a slash (/) in the input box and choose **/sdd-new** from the shortcut menu.
- Enter the task you want to complete (for example, writing a calculator app) and click  or press **Enter**.
- The agent generates a requirement specification document named **spec.md** based on the task you entered. If the generated **spec.md** file does not meet your expectations, you can manually modify it or use the agent to modify it.

Figure 3-12 Generating spec.md



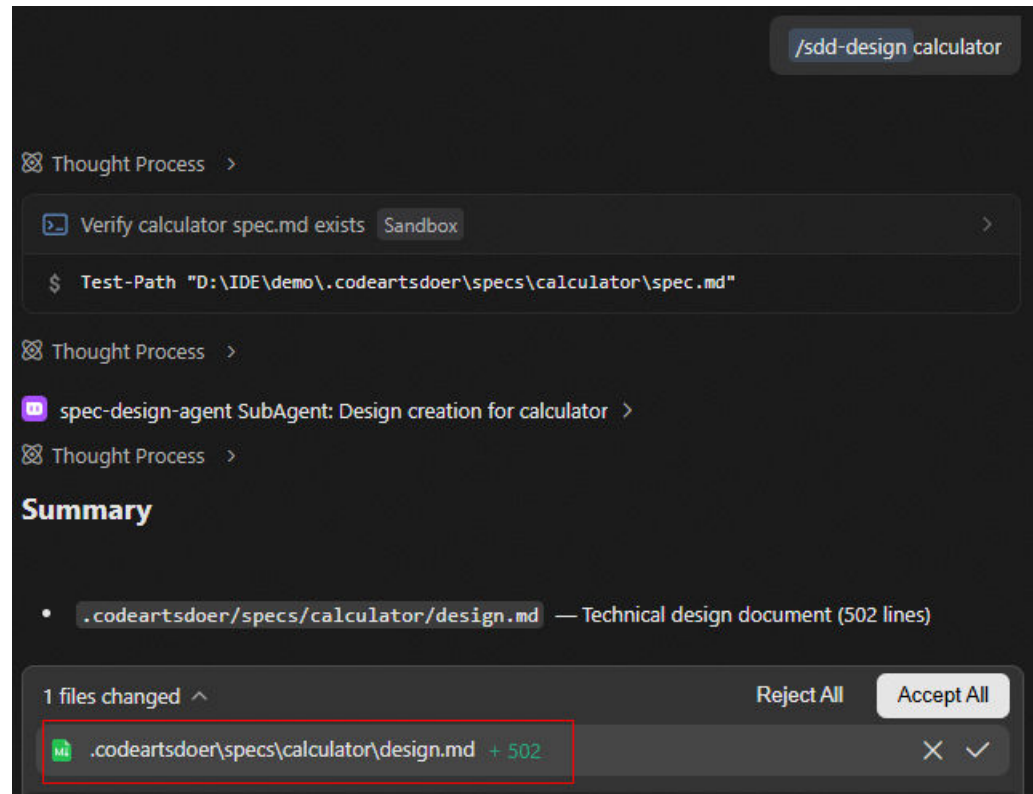
- Open *Project root directory* `/.codeartsdoer/specs/Name of the generated file`. You can find the generated **spec.md** file.

Step 3 Create a technical design.

Enter a slash (/) in the input box and choose **/sdd-design** from the shortcut menu. Then enter the artifact folder name (for example, calculator) and click  or press **Enter**.

The agent generates the **design.md** file based on the **spec.md** file obtained in the previous step. If the generated **design.md** file does not meet your expectations, you can manually modify it or use the agent to modify it.

Figure 3-13 Generate design.md



Step 4 Create a coding task list.

Enter a slash (/) in the input box and choose **/sdd-tasks** from the shortcut menu. Then enter the artifact folder name (for example, calculator) and click  or press **Enter**.

The agent generates the task list **tasks.md** based on **spec.md** and **design.md**. If the generated **tasks.md** file does not meet your expectations, you can manually modify it or use the agent to modify it.

Step 5 Perform coding tasks.

Enter a slash (/) in the input box and choose **/sdd-apply** from the shortcut menu. Then enter the artifact folder name (for example, **calculator**) and click  or press **Enter**.

The agent performs coding tasks item by item according to **tasks.md**.

----End

Specifying the Agent Running Mode

To ensure the security and flexibility of agent running, CodeArts Agent provides three running policies: sandbox running, automatic running, and manual running. You can select a mode based on your service requirements for security and

automation. **By default, the sandbox running mode is enabled for CodeArts Agent.**

To set the agent running mode, perform the following steps:

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
3. In the navigation pane on the left, choose **Chats**.
4. In the **Agents > Terminal Command Running Mode** area, specify the policy for the agent to run terminal commands.

Table 3-13 Parameters for setting the running mode

Parameter	Description
Terminal Command Running Mode	Specifies the policy for an agent to execute terminal commands. You can select a policy based on service requirements for security and automation. • Running in Sandbox: Operations are performed automatically in a secure sandbox by default. • Auto Running: The agent directly executes all commands without your approval. For security purposes, you are advised to select Auto Running only when necessary. In this mode, the agent bypasses all security checks and may perform high-risk operations without notifications. • Manual Running: Before executing any command, the agent sends a confirmation prompt to you. You need to manually confirm it.
Command Whitelist	This configuration item is displayed only when Terminal Command Running Mode is set to Running in Sandbox. Allows you to add the prefix of a specific command to the whitelist as required. Commands added to the whitelist bypass the sandbox mechanism and are executed outside the sandbox.
Custom Sandbox Settings	This configuration item is displayed only when Terminal Command Running Mode is set to Running in Sandbox. To configure the file system and network access rules for the sandbox environment, see Step 5.

Parameter	Description
Network Access	This configuration item is displayed only when Terminal Command Running Mode is set to Running in Sandbox. Allows you to choose from the provided network access policies based on actual needs and security levels. • Network-wide: Access to all internal and external network resources is allowed. • Local: Only the local network (LAN/intranet) can be accessed. Access to the Internet is not allowed. This option is unavailable for the Linux sandbox. • Blocked: All network connections are blocked, and access to internal or external resources is prohibited. In the Linux environment, the Linux sandbox generates an isolated virtual address (127.0.0.1), which is separated from the host machine's localhost. Even when network access is prohibited, this virtual address can still be accessed in the sandbox, and such access will not be blocked by the sandbox. • Custom: Allows you to customize the network access scope for processes within the project sandbox environment by configuring the sandbox.json file. For details, see Step 5.

Setting the Automatic Execution Permission

If the automatic execution permission is not enabled and the agent needs to call tools, you need to manually approve this operation, such as selecting **Run Once** or **Skip Unauthorized Tools**. After automatic execution is enabled, the related operations are automatically approved without your intervention. In addition, when you manually confirm the operation, you can also select **Ask Each Time** or **Always Allow and Enable Config** to meet different requirements.

CAUTION

Turning on automatic execution can lead to operational risks. Assess these risks carefully first and only enable this feature in secure and trustworthy environment.

To set the automatic execution permission, perform the following steps:

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. Click  in the upper right corner of CodeArts Agent IDE to open the settings page.

3. In the navigation pane on the left, choose **Chats**.
4. In the **Agents > Auto-approve area**, enable automatic execution if required. This will allow agents to perform the following operations without manual confirmation.

Table 3-14 Automatic operation items

Item	Description
Edit	Allows the agent to call tools like edit, write, and deleteFile to edit files on your computer.
Browser Access	Allows the agent to access websites in the browser.
Web Crawler	Allows the agent to access and capture specified web page content.

Optimizing Input

CodeArts Agent optimizes user input to help beginners ask questions more clearly and professionally, enabling users to obtain comprehensive, structured, and high-quality responses at one time. This effectively reduces repeated follow-up questions and enhances communication efficiency as well as user experience.

You can refer to the following steps to optimize your input:

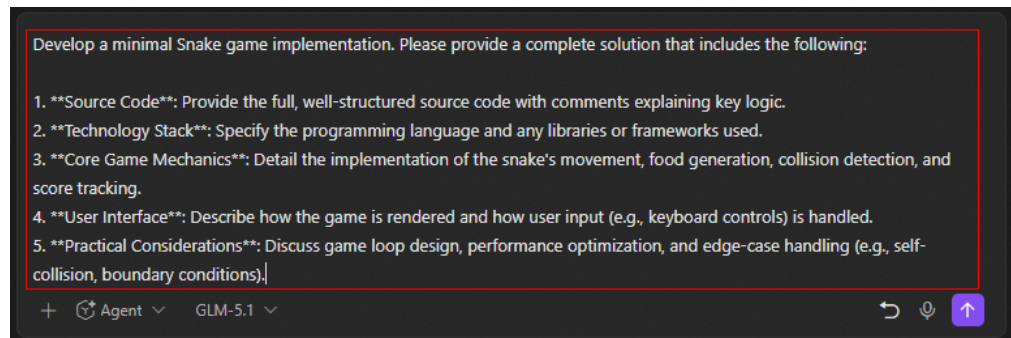
1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

3. Enter a prompt, for example, "Develop a mini snake game." Click  and wait the prompt optimization completes.

Figure 3-14 Optimizing input

Figure 3-15 After optimization

If the optimization result does not meet your expectations, click  to restore to the original prompt.

Unit Test (Agent)

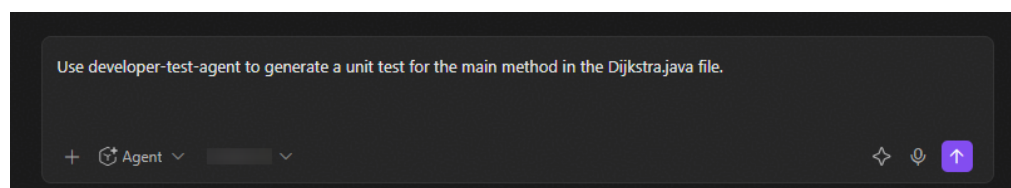
You can use commands to trigger the built-in developer-test-agent for code unit test. For example: Use the developer-test-agent to generate unit tests for a specific method within a file.

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. At the bottom of the input box on the chat panel, choose **Built-in** > **Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

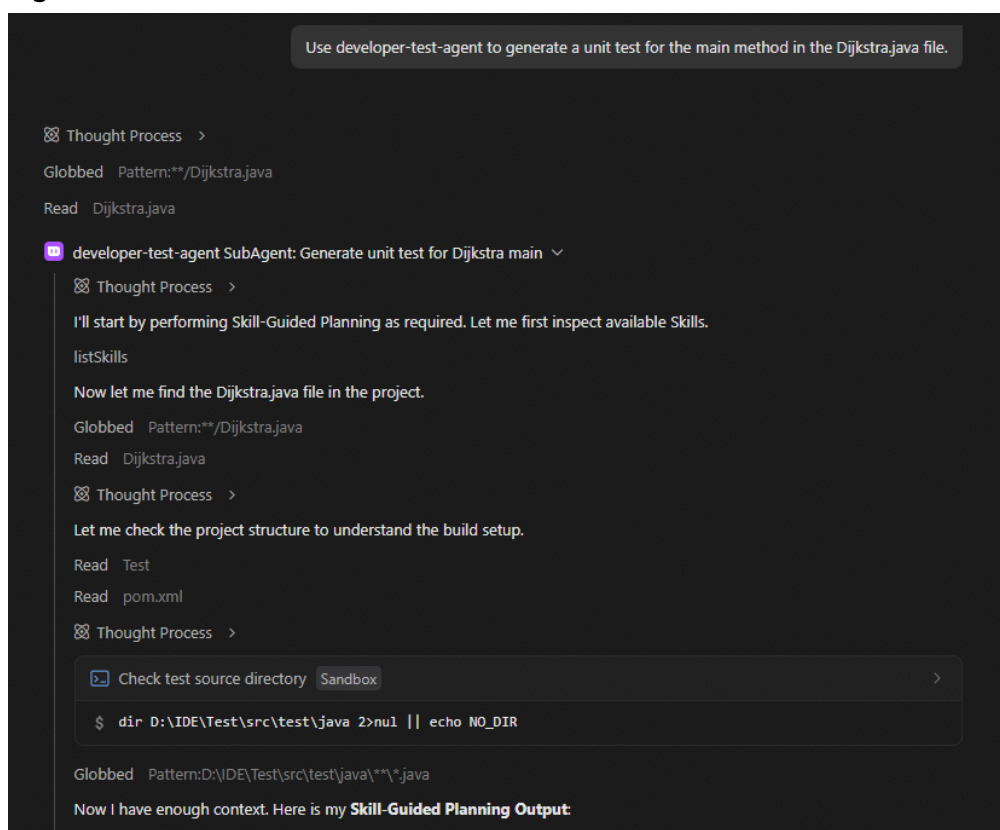
If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

3. Enter a prompt directly into the chat box to invoke it. For example, use developer-test-agent to generate a unit test for the main method in the **Dijkstra.java** file.

Figure 3-16 Unit test

4. Check whether a unit test file is generated. If yes, the execution is successful and the test is passed. The figure below is for reference only. The actual file may vary.

Figure 3-17 Successful test



Context

Give the agent details like specific files, folders, Git history, or terminal outputs. This helps them grasp your project setup and requirements clearly, minimizing errors and producing precise code.

For details, see [Context](#).

MCP

Agents can access external data sources and tools through MCP servers to handle various service needs.

For details, see [MCP](#).

Skills

A skill is a reusable module that encapsulates expertise like coding experience and common solutions. Agents can invoke these skills to dynamically load "skill packages" on demand, boosting efficiency and performance for specific tasks.

For details, see [Skills](#).

Diff View

Once the agent finishes generating or optimizing code, check the modifications in the diff view. It highlights added and removed lines with distinct colors, making it easy for you to review changes and confirm accuracy.

The diff view is automatically enabled. For details, see [Diff View](#).

Multi-Chat Management

In traditional development, tasks in a project must be executed one by one. Only one task can be executed at a time, resulting in low efficiency. With the multi-chat management function, the system can execute multiple tasks at the same time, improving the overall development efficiency.

For details, see [Multi-Chat Management](#).

Checkpoints

Checkpoint is a built-in mechanism that enables CodeArts Agent to automatically take snapshots of code status and allow for rollback. The system will automatically track the editing operations when the agent is running. Before every time that the agent modifies code, the system will automatically save the current code and chat status for quick cancellation, rollback, and historical version recovery. This function is applicable to AI-assisted development scenarios that require quick trial-and-error, secure editing, and one-click rollback.

For details, see [Checkpoints](#).

LSP

CodeArts Agent features a pre-installed local LSP function. Once enabled, the agent can leverage the diagnostic information, code definition, and symbol information provided by the language service, achieving more precise code understanding and modification.

For details, see [LSP](#).

3.4 Sandbox

A sandbox provides an **isolated and restricted execution space** for instructions generated by agents. Through strict permission control, it ensures that commands run within a securely isolated environment. This effectively blocks access to unauthorized resources (such as files and networks) or the execution of high-risk commands (such as permission modification commands). For intercepted commands, the agent will issue a risk prompt within a chat. They will only be executed outside the sandbox after a secondary confirmation by the user.

Using a sandbox offers the following advantages:

- **Preventing data leakage:** Based on sandbox technology, system-level and user-level sensitive directories are isolated, blocking data leakage risks at the source and comprehensively protecting core assets.
- **Preventing direct execution of high-risk commands:** Commands are executed in a sandbox environment, where potentially dangerous operations (e.g., deleting all files on the D drive or modifying user permissions) can be identified and blocked in advance. A prompt is also generated in the chat flow for users to confirm whether to execute the commands.
- **Supporting enterprise compliance requirements:** This mechanism meets stringent regulatory requirements for development environment file isolation

in industries such as finance, healthcare, and government, reducing security concerns and accelerating enterprise procurement decisions.

Constraints

Table 3-15 Constraints

Category	Description
OS	- Windows OS: Windows 11 (x64) is recommended. For Windows 10 (x64), the version must be 2019 or later, and upgrading to the latest stable version is advised. - macOS: macOS 11 or later, compatible with ARM64 (Apple Silicon). - Linux OS: Those on the x64 and ARM64 architectures are supported.
Controlled object	- Windows: Only Bash and PowerShell commands are controlled. - macOS: Only Shell commands are controlled. - Linux: Only Shell commands are controlled.

File Access Control

After sandbox is enabled, the file and directory access permissions for CodeArts Agent are configured as follows. You can also configure the **sandbox.json** file to customize the file and network access scope of processes in the sandbox environment of the current project. For details, see [Step 5](#).

Table 3-16 File access control

OS	Permission Type	Directory Type	Directory List
Windows	Read-only	-	All directories are readable except critical Windows system directories and sensitive user directories.
	Read/Write	Project directories and their subdirectories	-

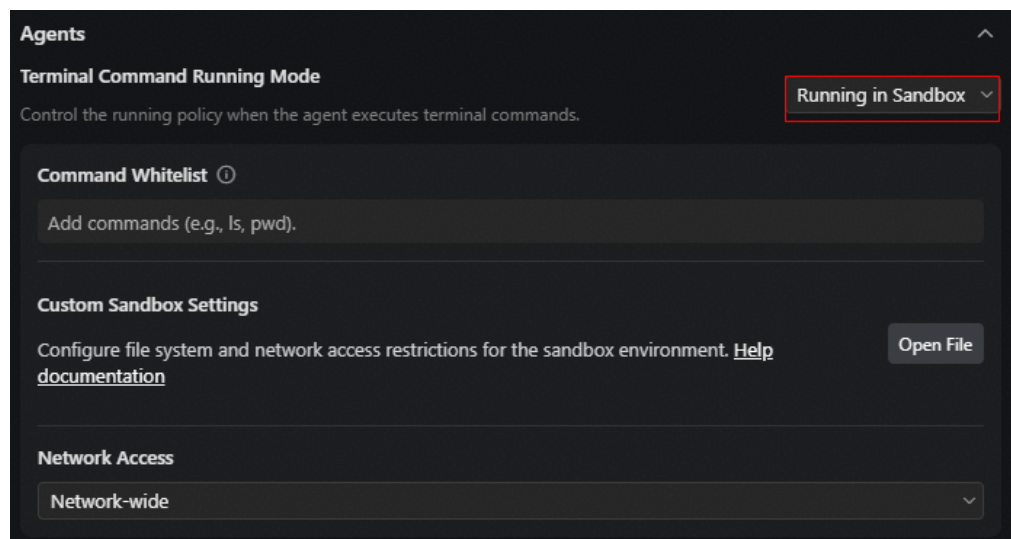
OS	Permission Type	Directory Type	Directory List
	No read/No write	Critical Windows system directories	• C:\Windows\System32\config • C:\Windows\System32\drivers\etc • C:\Windows\SysWOW64\config • C:\ProgramData\Microsoft\Crypto • C:\Windows\System32\GroupPolicy • C:\Windows\System32\GroupPolicyUsers • C:\ProgramData\Microsoft\Windows\WER
		Sensitive user directories	• C:\Users*\AppData\Local\Microsoft\Credentials • C:\Users*\AppData\Roaming\Microsoft\Credentials • C:\Users*\AppData\Local\Microsoft\Protect • C:\Users*\NTUSER.DAT • C:\Users*\ntuser.dat.LOG
macOS	Read-only	-	All directories are readable except sensitive system directories .
	Read/Write	Workspace and additionally configured directories	• Temporary directories: /tmp, /var/ folders, and TMPDIR environment variable paths • Cache directories: ~/Library/Caches and ~/.cache • Dependency directories of common tools: ~/.local/lib, ~/.local/bin, and ~/.local/share • Toolchain and dependency directories of common development languages (Go, Java, Python, Node.js, Rust, and Ruby)
	No read/No write	Documents/ Desktop/ Downloads (privacy-related)	• ~/Desktop • ~/Documents • ~/Downloads • ~/Pictures • ~/Movies • ~/.ssh • ~/.zsh_history

OS	Permission Type	Directory Type	Directory List
		Password/ Wallet/ Keychain- related	~/Library/Keychains
		System-level sensitive configuration	• /etc/passwd • /private/etc/passwd • /etc/group • /private/etc/group • /etc/hosts • /private/etc/hosts • /etc/resolv.conf • /private/etc/resolv.conf • /etc/pam.d • /private/etc/pam.d
Linux	Read-only	-	All directories are readable except sensitive system directories .
	Read/Write	Workspace and additionally configured directories	• Temporary directories: /tmp and TMPDIR environment variable paths • Cache directories: ~/.cache and XDG_CACHE_HOME environment variable paths • Dependency directories of common tools: ~/.local/lib, ~/.local/bin, and ~/.local/share • Toolchain and dependency directories of common development languages (Go, Java, Python, Node.js, Rust, and Ruby)
	No read/No write	Critical Linux system directories	• /etc/shadow • /etc/passwd • /etc/group • /etc/gshadow • /etc/resolv.conf • /etc/sudoers

Enabling the Sandbox Mode

- Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
- Step 3** Enable the sandbox mode.
1. In the navigation pane on the left, choose **Chats**.
 2. Under **Agents**, select **Running in Sandbox** from the **Terminal Command Running Mode** drop-down list.

Figure 3-18 Selecting Running in Sandbox



- Step 4** Configure the command whitelist.

This allows you to add the prefix of a specific command to the whitelist as required. Commands added to the whitelist bypass the sandbox mechanism and are executed outside the sandbox.

- Step 5** Customize sandbox configuration.

By configuring the **sandbox.json** file, you can customize files and the network access scopes for processes within the current project's sandbox environment.

1. Click **Open Configuration** next to **Sandbox Custom Configuration**. The IDE will generate a **sandbox.json** file in your local user directory (**%USERPROFILE%/.codeartsdoer**) and automatically open it in the editor.

The initial structure of the file is as follows:

```
{
  "filesystem": {
    "readWrite": [],
    "readOnly": []
  },
  "network": {
    "default": "Allow",
    "allow": [],
    "deny": []
  },
  "resources": {
```



```
"cpu": 50,  
"memory": 2  
}  
}
```

Table 3-17 Parameters in the sandbox.json file

Parameter	Mandator y (Yes/No)	Parameter Type	Description
filesystem	No	filesystem Object	Used to precisely control the sandbox's access permissions to the local file system. If it is not set (the filesystem field is empty or does not exist), the sandbox's built-in file system security policy will be applied.
network	No	network Object	Used to control the network access policies for processes within the sandbox, supporting configurations to allow or block access to specific network resources. If it is not set, network access is allowed by default.
resources	No	resources Object	Used to define the maximum limit of computing resources during sandbox runtime, ensuring service stability and preventing resource abuse. If it is not set, the system will share the host machine's resources.

Table 3-18 Parameters in filesystem

Parameter	Parameter Type	Default Value	Format of Supported Paths	Priority Rule	Description
readWrite	Array	[]	- Absolute path: for example, /home/user/project or C:\Projects - Relative path: for example, ./src or ./config - Environment variable: \$HOME (Linux/Mac) or %USERPROFILE% (Windows) - Abbreviation of the home directory: ~	readOnly > readWrite > Default system policy If a specific path matches both readOnly and readWrite, readOnly shall prevail.	List of read/write paths.
readOnly	Array	[]			List of read-only paths.

Table 3-19 Parameters in network

Parameter	Parameter Type	Default Value	Priority Rule	Description
default	String	Allow	deny > allow > default If both allow and deny are configured, deny shall prevail.	Default network access policy. - Allow: access allowed. - Deny: access denied. NOTE This field supports two configuration formats: <i>Domain:Port</i> and <i>IP address:Port</i> . Wildcards are supported for the domain part, and CIDR notation is supported for IP addresses. Multiple ports can be configured by separating them with commas. If no port is specified, the rule applies to all ports by default.
allow	Array	[]		Network rules that allow access to processes in the sandbox.
deny	Array	[]		Network rules that reject access to processes in the sandbox.

Table 3-20 Parameters in resources

Parameter	Parameter Type	Default Value	Min. Value	Description
cpu	Integer	50	20	CPU usage, in percentage.
memory	Integer	8	1	Memory size, in GB.

2. In the opened **sandbox.json** file, enter your custom sandbox configuration.

Example configuration of **sandbox.json**:

```
{
  "filesystem": {
    "readWrite": [
      "/home/user/project/output",
      "~/workspace/temp"
    ],
    "readOnly": [
      "/etc/systemd",
      "%USERPROFILE%/.ssh"
    ]
  },
  "network": {
    "default": "Allow",
    "deny": [
      "10.0.0.0/8",
      "192.168.0.0/16"
    ]
  },
  "resources": {
    "cpu": 50,
    "memory": 4
  }
}
```

3. Save the file for the configuration to take effect.

Step 6 Configure the network access policy.

Table 3-21 Network access policy description

Parameter	Description
Network-wide	Allows access to all internal and external networks.
Local	Only the local network (LAN/intranet) can be accessed. Access to external networks is not allowed. This option is unavailable for the Linux sandbox.
Blocked	All network connections are blocked, and access to any internal or external network is prohibited. In the Linux environment, the Linux sandbox generates an isolated virtual address (127.0.0.1), which is separated from the host machine's localhost. Even when network access is prohibited, this virtual address can still be accessed in the sandbox, and such access will not be blocked by the sandbox.
Custom	Allows you to modify the JSON policy configuration file to customize the network access scope for processes within the project sandbox environment. For details about how to configure the JSON file, see Step 5 .

Step 7 Exit the current setting page. The sandbox mode is enabled.

----End

Execution Policy for High-Risk Commands

When the agent detects a high-risk command, the AI will display a prompt in the chat flow. You need to evaluate risks and select an execution mode as required.

- **Skip:** The target command is not executed.
- **Trust XXX:** The prefix of the target command is added to the whitelist. Then all commands starting with this prefix will be executed outside the sandbox.
- **Continue running:** The target command can be executed outside the sandbox for the current chat.

3.5 Multi-Chat Management

In traditional development, tasks in a project must be executed one by one. Only one task can be executed at a time, resulting in low efficiency. With the multi-chat management function, the system can execute multiple tasks at the same time, improving the overall development efficiency.

Constraints

Table 3-22 Constraints

Category	Description
Quantity	For a user, a maximum of five chats can be created concurrently across all tools (such as CodeArts Agent IDE and Visual Studio Code).
Mode	Multi-chat management is supported only for the vibe coding when Agent or a custom agent is selected.

Creating Multiple Chats

Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

Step 2 At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

Step 3 Create the first session.

Type your task into the box, and click  or press **Enter** to submit your request.

Step 4 While the first chat is ongoing, click  in the upper right corner of the chat panel to open a new chat. The first chat will continue to run in the background.

Step 5 Create the second session.

Type your task into the box, and click  or press **Enter** to submit your request.

A user can create and run a maximum of **five** chats concurrently across all tools (such as CodeArts Agent IDE and Visual Studio Code).

----End

Managing Multiple Chats

Table 3-23 Managing Multiple Chats

Operation	Procedure
Viewing the chat progress	In the upper right corner of the chat panel, click . In the displayed recent chats, view their progress. : The chat is running. : The chat is waiting for user confirmation. : The chat is completed.
Switching between chats	In the upper right corner of the chat panel, click . Click the name of the chat to be viewed.
Stopping a chat	- Stopping the current session: Click  in the lower right corner of the chat panel to stop the chat. - Stopping a background chat: In the recent chat list, click the target chat. Click  in the lower right corner of the chat panel to stop the chat.
Deleting a chat	- In the upper right corner of the chat panel, click . - Hover the cursor over  next to the chat to be deleted and click Delete in the displayed list. A confirmation dialog box is displayed. - Click OK to delete the chat. If the chat is removed from the recent chat list, it has been deleted.

3.6 Checkpoints

Checkpoint is a built-in mechanism that enables CodeArts Agent to automatically take snapshots of code status and allow for rollback. The system will automatically track the editing operations when the agent is running. Before every time that the agent modifies code, the system will automatically save the current code and chat status for quick cancellation, rollback, and historical version recovery. This function is applicable to AI-assisted development scenarios that require quick trial-and-error, secure editing, and one-click rollback.

Constraints

Table 3-24 Constraints

Category	Description
Time	Checkpoints are valid for 15 days and will be automatically cleared upon expiration. Once they expire, the rollback button will no longer be displayed, and historical versions cannot be restored.
Model	This feature is available only when Agent or a custom agent is selected.
File size and type	Checkpoints do not track binary files such as images, videos, archives, EXEs, and PDFs, and they ignore text files exceeding 10 MB. For text files larger than 2 MB, a diff view will not be available during version rollback previews.
Function	• The system automatically generates a checkpoint before each AI edit, tracking only file modifications made by AI editing tools within the current chat. • File modifications made via Bash commands or external editors will not be tracked. • AI modifications made to the same file across multiple chats are consolidated into that file's checkpoint. Modifications to different files are isolated from each other. However, manual modifications by users or changes made by other software are not tracked by the system and will not trigger checkpoint generation. • Checkpoints are designed to quickly undo or roll back modifications within the current chat, which is temporary recovery. They are not a substitute for professional version control systems like Git. • For files outside the project, checkpoints only support rolling back the chat content, not the file content.

Automatic Tracking

The automatic tracking capabilities of CodeArts Agent are as follows:

- **Tracking all edits:** Modifications made by AI file editors are automatically tracked.
- **Creating checkpoints based on prompts:** Each time a user submits a prompt, the system creates a checkpoint before AI edits, recording the current code state.
- **Persistence:** Users can choose to restore their code to a specific checkpoint within historical chats. Each checkpoint remains valid for 15 days.

Rolling Back a Chat

Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

Step 2 At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

Step 3 On the chat panel of CodeArts Agent, roll back a chat.

CAUTION

The rollback operation cannot be undone. Exercise caution when performing this operation.

Rolling Back a Specified Chat in Historical Chats

1. Click  in the upper right corner.
2. Click the name of the chat to be rolled back.
3. Hover the cursor over the chat to be rolled back, and click . A confirmation dialog box is displayed.
 - If a file is modified, the name of the file to be restored is displayed.
 - If no file is modified, no file information is displayed.

Figure 3-19 Confirmation dialog box for restoration (files modified)

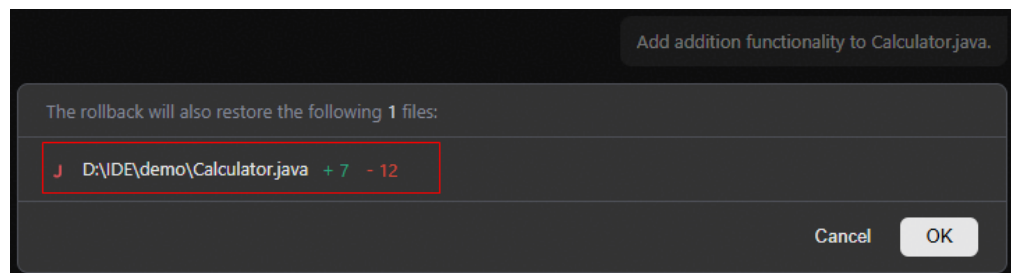
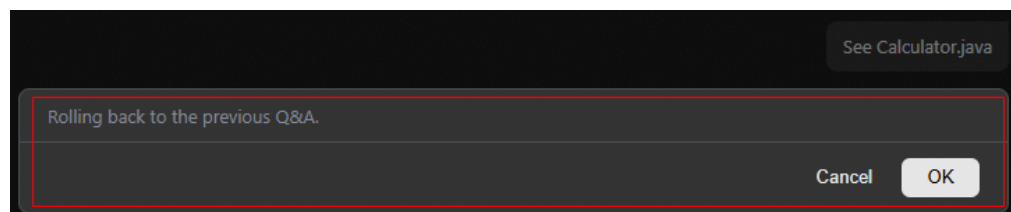
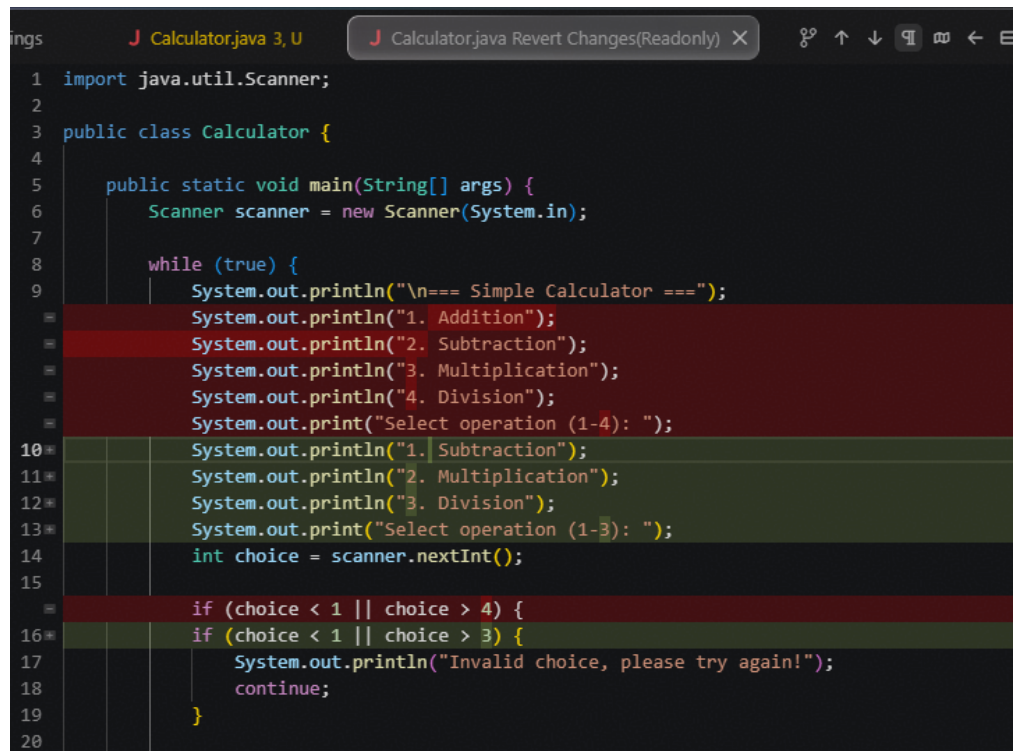


Figure 3-20 Confirmation dialog box for restoration (no files modified)



4. (Optional) If files have been modified, click the name of the file you want to restore to open the diff view before and after the rollback in the editor area.

Figure 3-21 Diff view



```
1 import java.util.Scanner;
2
3 public class Calculator {
4
5     public static void main(String[] args) {
6         Scanner scanner = new Scanner(System.in);
7
8         while (true) {
9             System.out.println("\n=== Simple Calculator ===");
10            System.out.println("1. Addition");
11            System.out.println("2. Subtraction");
12            System.out.println("3. Multiplication");
13            System.out.println("4. Division");
14            System.out.print("Select operation (1-4): ");
15            System.out.println("1. Subtraction");
16            System.out.println("2. Multiplication");
17            System.out.println("3. Division");
18            System.out.print("Select operation (1-3): ");
19            int choice = scanner.nextInt();
20
21            if (choice < 1 || choice > 4) {
22                if (choice < 1 || choice > 3) {
23                    System.out.println("Invalid choice, please try again!");
24                    continue;
25                }
26            }
27        }
28    }
29 }
```

5. Click **OK** to roll back the chat.
 - If files have been modified, they will be rolled back to their historical states, and the messages at and after this timestamp in the current chat will disappear.
 - If no files have been modified, the messages at and after this timestamp in the current chat will disappear.

Rolling Back the Current Chat

1. Hover the cursor over the chat to be rolled back, and click . A confirmation dialog box is displayed.
 - If a file is modified, the name of the file to be restored is displayed.
 - If no file is modified, no file information is displayed.

Figure 3-22 Confirmation dialog box for restoration (files modified)

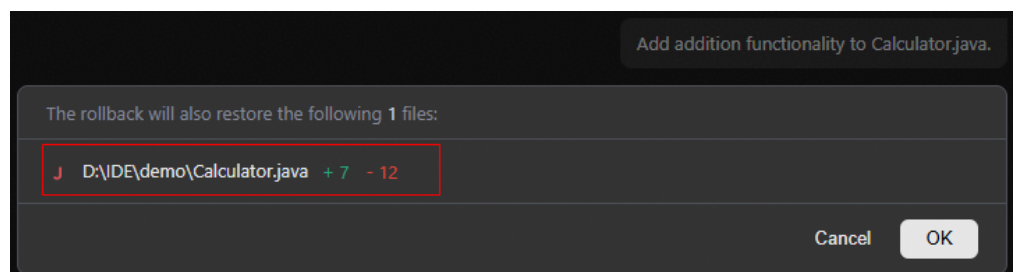
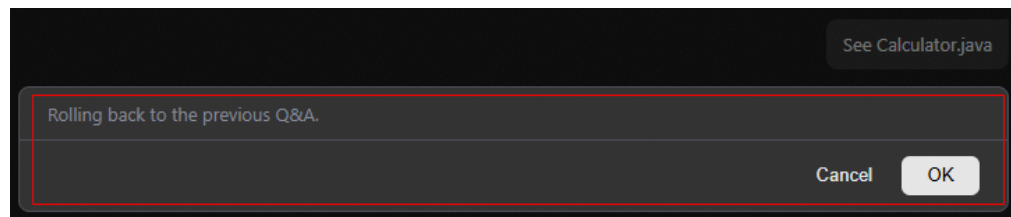
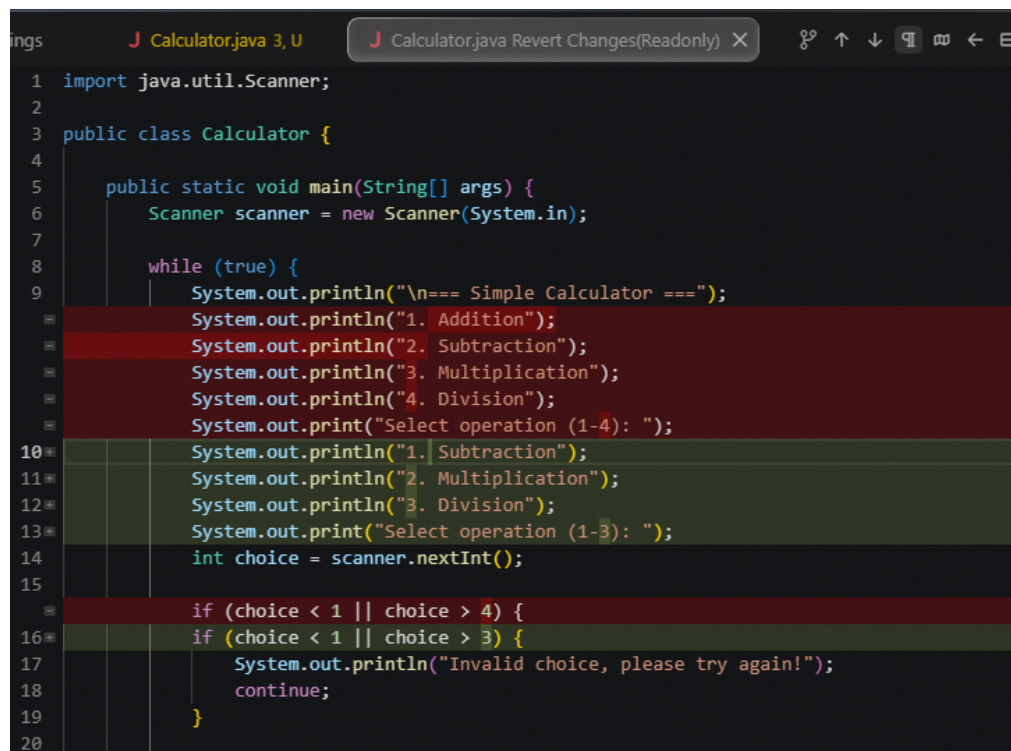


Figure 3-23 Confirmation dialog box for restoration (no files modified)

2. (Optional) If files have been modified, click the name of the file you want to restore to open the diff view before and after the rollback in the editor area.

Figure 3-24 Diff view

3. Click **OK** to roll back the chat.
 - If files have been modified, they will be rolled back to their historical states, and the messages at and after this timestamp in the current chat will disappear.
 - If no files have been modified, the messages at and after this timestamp in the current chat will disappear.

----End

Related Documents

If you need an end-to-end example of checkpoints, see Best Practices for Checkpoint Session Rollback. This practice uses a lightweight coffee ordering system to demonstrate checkpoint capabilities during chatting, including status archiving, precise rollback, and UI restoration.

3.7 Diff View

Within your editor, CodeArts Agent shows all code changes clearly in its diff view, whether you create a new file or edit an existing one. You can quickly check details of each change before and after applying them, keeping adjustments transparent and manageable.

Constraints

This feature is available only when **Agent** or a custom agent is selected.

Diff Preview

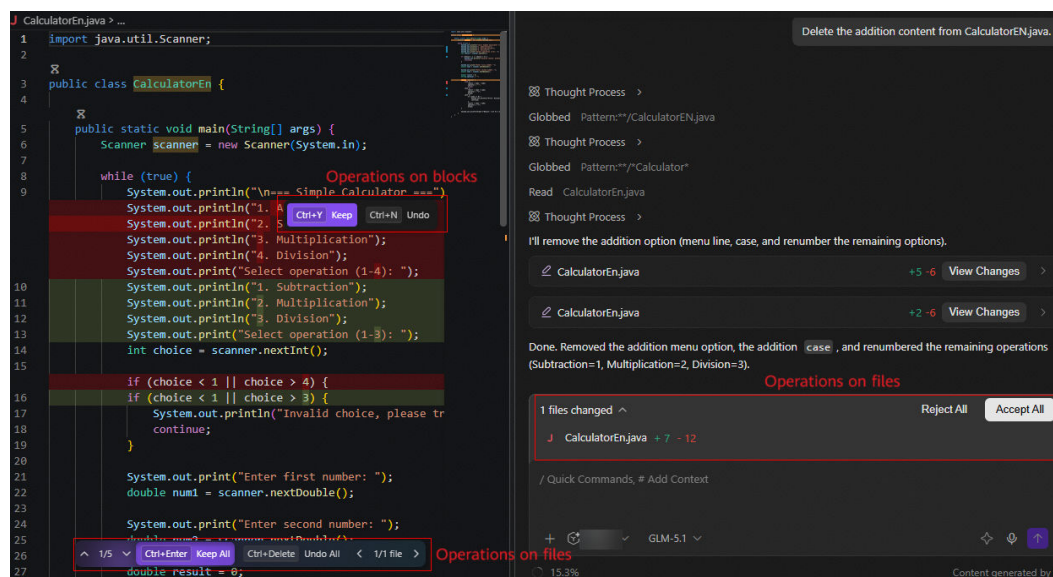
- Green highlights mark new lines.
- Red highlights mark deleted lines.

Diff View

The agent's diff view shows context changes clearly for better understanding. On the chat page, you will see the total files changed, line counts, and a file list. Click **View Change** beside any file to check its update details in the diff view.

- Operations on files: Click **Keep All** (Ctrl+Enter) or **Undo All** (Ctrl+Delete) at the bottom of the editing area, or choose **Accept All** or **Reject All** above the chat box to apply or reject all modifications at once. Hover over a file and click  or  to instantly approve or decline its changes.
- Operations on blocks: In the editing area, click **Undo** (Ctrl+N) for the target code to restore it to the original state. If the modification meets the expectation, click **Keep** (Ctrl+Y) to apply the modification.

Figure 3-25 Display effect of the Diff view



3.8 LSP

Language Server Protocol (LSP) is a standardized protocol for providing language services. By decoupling language services from the IDE, LSP enables an independent language server process to deliver core capabilities such as code analysis, fast navigation, and symbol search.

CodeArts Agent now features a built-in local LSP function. Once enabled, the agent can leverage these language services to provide code analysis, code navigation, and symbol information, enabling more accurate code comprehension and modification.

Constraints

This function can be used only in **agent** mode.

Supported Mainstream LSP Servers

Table 3-25 Supported mainstream LSP servers

LSP Server	Extension Name	Requirement
jdts	.java	Ensure that Java SDK 21 or later has been installed.
typescript-language-server	.ts, .tsx, .js, .jsx, .mjs, .cjs, .mts, and .cts	Ensure that TypeScript has been installed.
pyright	.py, and .pyi	Ensure that Pyright has been installed.
gopls	.go	Ensure that the Go language environment and go commands are available and configured in the system environment variables.

Enabling Local LSP

Step 1 Switch to the agent mode.

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

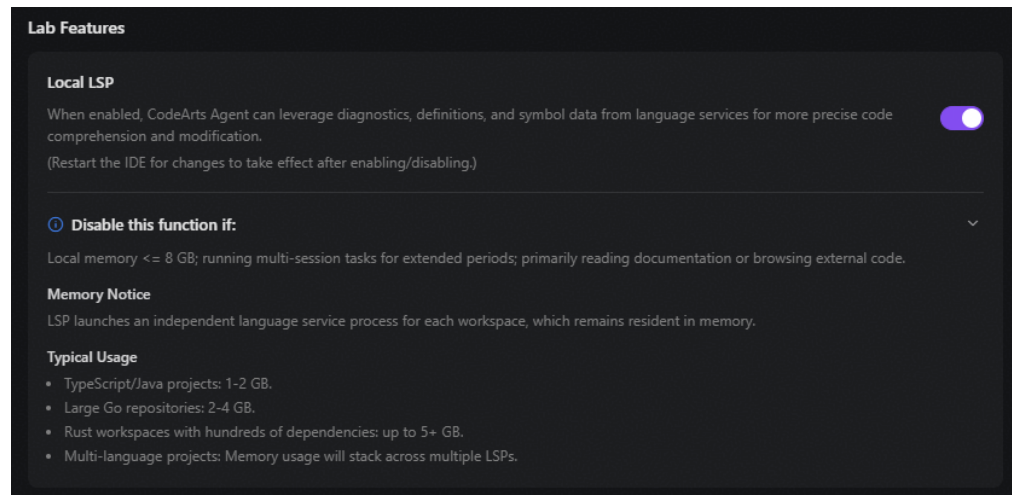
If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

Step 2 Enable local LSP.

1. Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
2. In the navigation pane on the left, choose **Lab Features**.
3. Enable local LSP.

Once enabled, the agent can leverage these language services to provide code navigation and symbol information, enabling more accurate code comprehension and modification.

Figure 3-26 Enabling a lab feature



Step 3 Restart the IDE for the change to take effect.

After CodeArts Agent IDE is restarted, **LSP service running** is displayed in the lower right corner of the page.

----End

Related Documents

After the local LSP service is enabled, if a message indicating that the LSP service is unavailable is displayed during an agent session, locate and rectify the fault by referring to the operations in [What Should I Do If the Service Is Unavailable After Enabling the Local LSP Function?](#)

4 Models

Enterprise members can add third-party LLMs to the CodeArts Agent client. If you have purchased CodeArts Agent professional edition, you can also add third-party LLMs on the console. For details, see "Configuring Custom Models".

Constraints

- Only third-party custom enterprise models using the OpenAI-compliant Chat Completions APIs and Anthropic-compliant Messages APIs can be integrated.
- All agents can invoke the custom enterprise models created on the console. However, only the built-in agents, AgentTeam, and custom agents can invoke the custom enterprise models created on the client, and intelligent Q&A cannot invoke them.
- The provider and model ID combination of each model you add must be unique.

Prerequisites

- Your account has been added as an enterprise member and assigned a seat.
- If you have purchased CodeArts Agent professional edition, ensure that the enterprise administrator has enabled model customization for members on the console. For details, see "Configuring Custom Models".

Viewing Built-in Models

Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

Step 2 Click  in the upper right corner of CodeArts Agent IDE to open the settings page.

Step 3 In the navigation pane of the **Settings** page, choose **Model**.

Step 4 View the built-in models in the **Built-in** area.

If you have purchased CodeArts Agent professional edition package, custom models created by the enterprise administrator on the console are also displayed in addition to the built-in ones. The service provider is Huawei Cloud CodeArts Agent for built-in models and system for custom models.

----End

Adding Custom Models

CodeArts Agent allows you to add custom models by specifying the model provider or using custom configurations.

Adding a Model by Specifying the Model Provider

- Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
- Step 3** In the navigation pane of the **Settings** page, choose **Model**.
- Step 4** Click **Add Model**.
- Step 5** Select **Model Provider**, and set parameters by referring to [Table 4-1](#).

Figure 4-1 Adding a model by specifying the model provider

Add Model

Model Provider

Custom Configuration

Model Provider

Huawei Cloud MaaS

Model ID

GLM5.1

API Key

Please enter API Key

Get API Key

Advanced Configuration

Custom Model URL

Custom Edit

https://api.modelarts-maas.com/v2

Custom Model Name

GLM5.1

6/64

Context Window Tokens

Input

184,000

Output

16,000

Description

Zhipu GLM-5.1 model, delivering excellent performance on Chinese natural language processing tasks

98/200

Disclaimer

CodeArts does not take responsibility for the availability, compliance, and security of third-party models. Please evaluate the relevant models and review their license agreements before use to ensure compliance with regulatory requirements.

Cancel

Confirm

Table 4-1 Adding a model by specifying the model provider

Parameter	Description
Model Provider	Select the provider to which the model to be added belongs from the drop-down list.
Model ID	Select the ID of the model to be added. If the required model ID is not in the list, select Custom .
Custom Model ID	This parameter is available only when Model ID is set to Custom . Enter the model ID obtained from the model provider.

Issue 01 (2026-07-29)

Copyright © Huawei Cloud Computing Technologies Co., Ltd.

56

Parameter	Description
API Key	Enter the API key obtained from the model provider.
Custom Model URL	Optional. After you select a provider, the system automatically populates the model URL. To modify the model URL, enable Custom Edit and enter the new model URL. Enter only the base URL of the API. For example, if the model address is https://open.bigmodel.cn/api/paas/v4/chat/completions , enter https://open.bigmodel.cn/api/paas/v4 here.
Custom Model Name	Optional. Enter the name customized for the model. The value contains a maximum of 64 characters.
Context Window Tokens	Optional. Set the input tokens and output tokens. • Input: indicates the maximum number of tokens that can be received by the model in a single request. • Output: indicates the maximum number of tokens that can be delivered by the model in a single response.
Description	Optional. Enter a maximum of 200 characters to describe the custom model.

Step 6 Click **OK**.

View the added custom model in the **Custom** area.

----End

Adding a Model Through Custom Configuration

Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

Step 2 Click  in the upper right corner of CodeArts Agent IDE to open the settings page.

Step 3 In the navigation pane of the **Settings** page, choose **Model**.

Step 4 Click **Add Model**.

Step 5 Select **Custom Configuration**, and set parameters by referring to [Table 4-2](#).

Figure 4-2 Adding a model through custom configuration

Add Model

Model Provider

Custom Configuration

API Format

OpenAI Chat Completions

Model URL

Please enter model URL

Please enter the OpenAI API endpoint address, do not end with a slash (/). The system will automatically append the /chat/completions path.

Model ID

Please enter model ID

0/64

API Key

Please enter API Key

Advanced Configuration

Custom Model Name

Please enter model display name, e.g.: OpenAI-3-Next

0/64

Context Window Tokens

Input184,000

Output16,000

Description

Please enter

0/200

Disclaimer

CodeArts does not take responsibility for the availability, compliance, and security of third-party models. Please evaluate the relevant models and review their license agreements before use to ensure compliance with regulatory requirements.

Cancel

Confirm

Table 4-2 Adding a model through custom configuration

Parameter	Description
API Format	Select the API format used by the model from the drop-down list. The options include OpenAI Chat Completions and Anthropic Messages .
Model URL	Enter the model URL obtained from the model provider. Enter only the base URL of the API. For example, if the model address is https://open.bigmodel.cn/api/paas/v4/chat/completions , enter https://open.bigmodel.cn/api/paas/v4 here.
Model ID	Enter the model ID obtained from the model provider. Its value contains a maximum of 64 characters. For example, glm-5.1 .
API Key	Enter the API key obtained from the model provider.

Parameter	Description
Custom Model Name	Optional. Enter the name customized for the model. The value contains a maximum of 64 characters.
Context Window Tokens	Optional. Set the input tokens and output tokens. • Input: indicates the maximum number of tokens that can be received by the model in a single request. • Output: indicates the maximum number of tokens that can be delivered by the model in a single response.
Description	Optional. Enter a maximum of 200 characters to describe the custom model.

Step 6 Click **OK**.

View the added custom model in the **Custom** area.

----End

Using Custom Models

In the lower part of the chat box, choose **Built-in > Agent**, or choose **Custom** and select a custom model from the model list on the right.

Managing Custom Models

After creating custom models, enterprise members can edit, delete, and disable them as required.

Operation	Procedure
Editing a custom model	1. In the Custom area, click  next to the target custom model. 2. Edit the model parameters and click OK.
Deleting a custom model	1. In the Custom area, click  next to the target custom model. 2. Click Delete. The target custom model is removed from the Custom area, indicating that it is successfully deleted.
Enabling/ Disabling a custom model	In the Custom area, click  or  next to the target custom model to enable or disable it. Once disabled, custom models will no longer show up in the model list in the lower part of the chat box.

5 Code Generation

CodeArts Agent can generate context-based code suggestions in the editor. **Code generation** can be triggered either **automatically** or through **shortcut keys**.

Automatic Code Generation

- While typing, pause to allow the tool to analyze the context and provide code suggestions.
- Press **Enter** to allow the tool to analyze the context and provide code suggestions.

Code Generation by Shortcut Keys

You can use shortcut keys to quickly activate the intelligent assistance function. CodeArts Agent automatically identifies the context and intelligently generates code snippets based on the code structure, variable types, and logical relationships.

Table 5-1 Shortcut keys

Shortcut Key		Description
Windows/ Linux	Mac	
Alt+X	Option(⌘)+X	Generate a code line.
Alt+C	Option(⌘)+C	Generate multiple code lines.
Tab	Tab	Accept the generated code.
Ctrl+Right	Command(⌘)+Right	Accept the generated code word by word.
Escape	Escape	Cancel the generated code.
Alt+R	Option(⌘)+R	Regenerate the code.
Alt+[	Option(⌘)+[	Go to the previous suggestion during code regeneration.

Shortcut Key		Description
Windows/ Linux	Mac	
Alt+]	Option(⌘)+]	Go to the next suggestion during code regeneration.

Example: Completing a Function Body

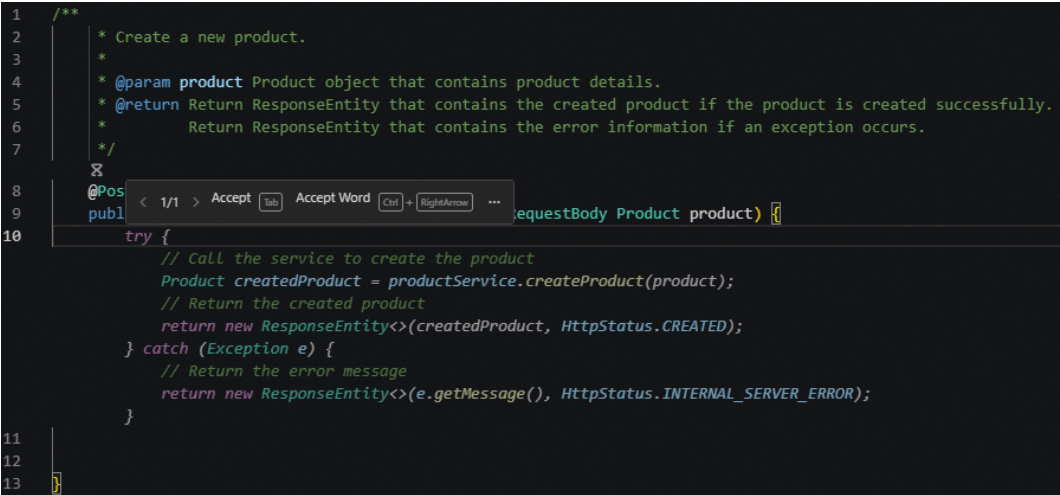
- Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2** In the editor, move the cursor to the end of the following method that needs to be completed:

```
/**
 * Create a new product.
 *
 * @param product Product object that contains product details.
 * @return ResponseEntity that contains the created product if the product is created successfully.
 *         Return ResponseEntity that contains the error information if an exception occurs.
 */
@PostMapping
public Class ResponseEntity<?> createProduct(@RequestBody Product product) {
}
```

- Step 3** Press **Enter**. CodeArts Agent will automatically complete the function body.

You can accept or reject the generated code suggestion. If the initial generation is incomplete, press **Tab** to accept what exists and then press **Alt+C** (Windows/Linux) or **Option (⌘)+C** (macOS) to continue the generation. Repeat this action until the code block is fully fleshed out.

Figure 5-1 Code generation



```
1  /**
2   * Create a new product.
3   *
4   * @param product Product object that contains product details.
5   * @return ResponseEntity that contains the created product if the product is created successfully.
6   *         Return ResponseEntity that contains the error information if an exception occurs.
7   */
8   @PostMapping
9   public ResponseEntity<?> createProduct(@RequestBody Product product) {
10      try {
11          // Call the service to create the product
12          Product createdProduct = productService.createProduct(product);
13          // Return the created product
14          return new ResponseEntity<>(createdProduct, HttpStatus.CREATED);
15      } catch (Exception e) {
16          // Return the error message
17          return new ResponseEntity<>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
18      }
19  }
```

-----End

6 Intelligent Q&A

During code development, you can ask CodeArts Agent questions at any time. CodeArts Agent quickly searches for R&D knowledge and returns answers accurately, helping you efficiently solve development problems.

Side Panel

If you face a coding issue, talk to CodeArts Agent in the side panel for quick solutions.

Step 1 Switch to the chat mode.

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. At the bottom of the input box on the chat panel, choose **Built-in > Ask** to switch to the intelligent Q&A mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

Step 2 Start a chat.

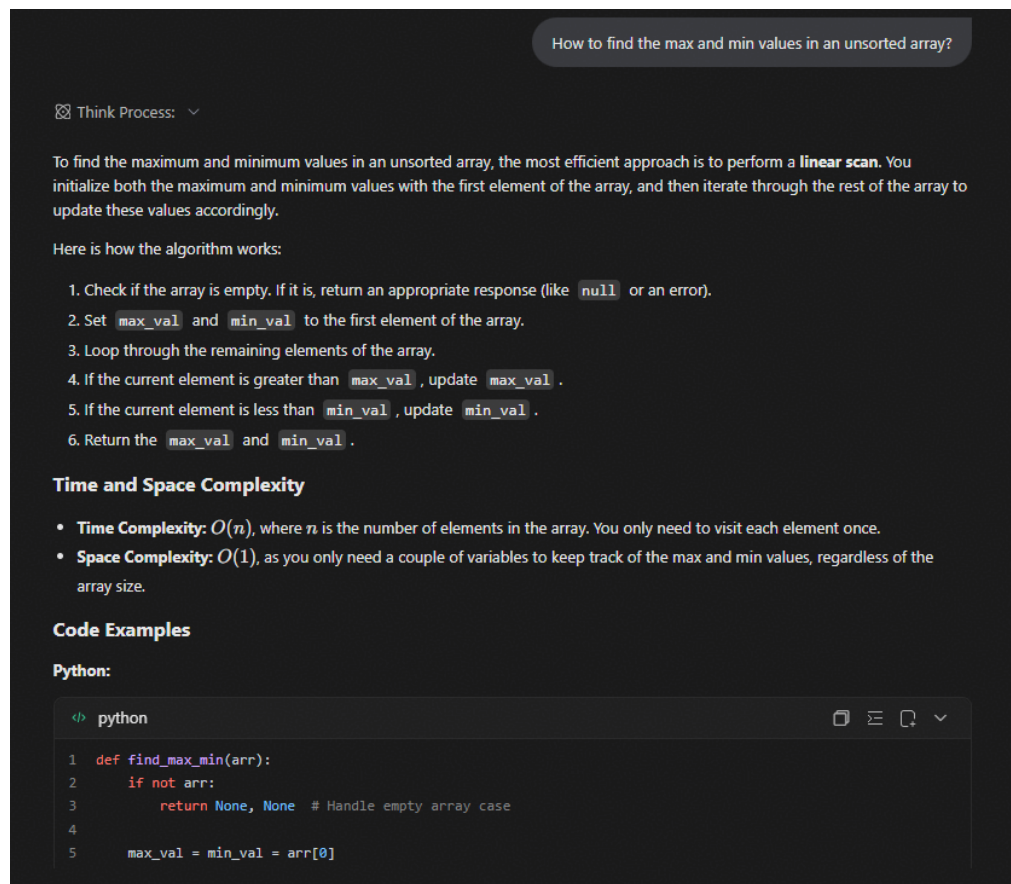
NOTE

During the generation of a response, the input box is unavailable. Wait until the response is completed and then send a new question.

Method 1: Type a question in the chat box.

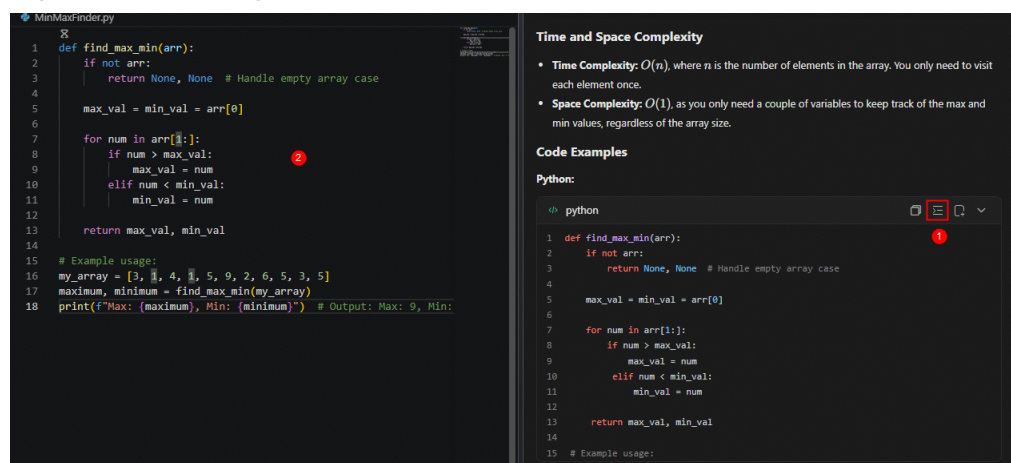
1. Type "How to find the max and min values in an unsorted array?" into the chat box.
2. Click . CodeArts Agent not only lists the programming ideas, but also provides the corresponding code implementation process.

Figure 6-1 Response



3. Create a **MinMaxFinder.py** file in the directory of the explorer and insert code.

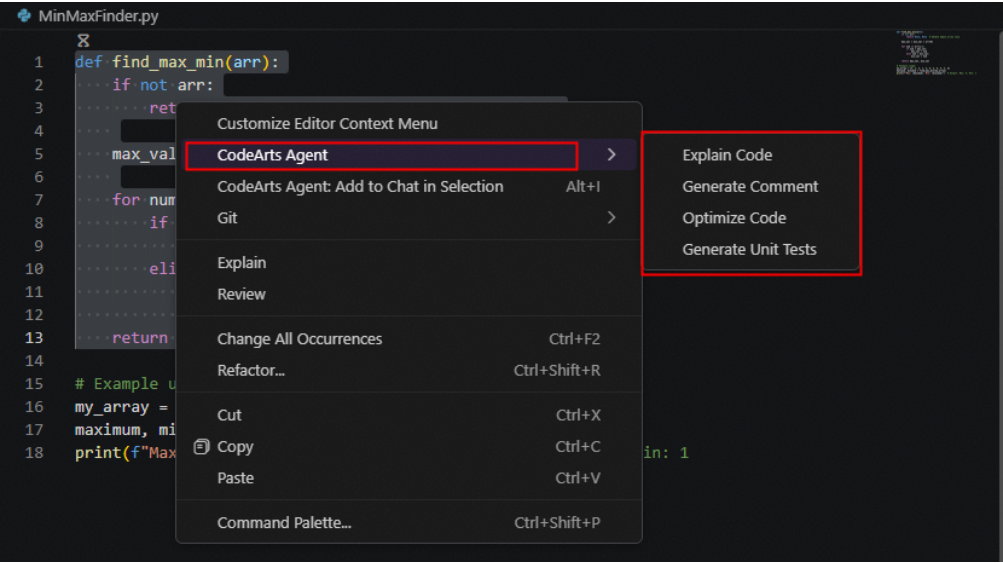
Figure 6-2 Inserting code



Method 2: Right-click in the code editor for quick access to actions.

1. In the code editor, select the code you need assistance with.
2. Right-click to expand the shortcut menu and select the desired operation.

Figure 6-3 Right-clicking in the editing area



Method 3: Enter / or choose + > **Shortcuts** bring up the shortcut menu.

1. In the input box on the chat panel, enter / or choose  > **Shortcuts** to bring up the shortcut menu.
2. Select the required item and click .

Table 6-1 Command types

Command	Description
/optimize	Optimizing code Allows CodeArts Agent to analyze code performance bottlenecks and provide optimizations, helping users improve the readability and maintainability of code.
/comment	Code commenting Allows the system to analyze the code's architecture and logic, then auto-generate clear, meaningful comments. It facilitates standardizing the comment style across your entire repository, so readability becomes second nature and code archaeology is a thing of the past.
/explain	Explaining code Allows the system to walk you through complex code with function, purpose, application scenario, and main logic. This clear explanation helps you get up to speed faster, and make knowledge flows smoothly, keeping the whole team in sync.

Command	Description
/tests	Unit test Allows the system to automatically generate high-quality test cases based on a single method and class, safeguarding code quality and effectively reducing defect risks.
/help	Help See shortcut operation tips.
/clear	Clear screen Clears the Q&A window content and opens a new one.

----End

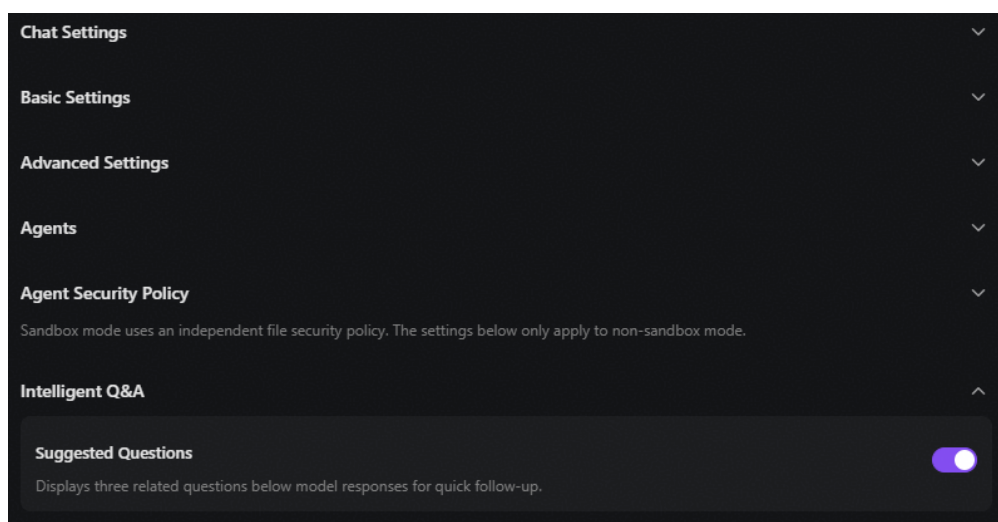
Suggested Questions

After you enable this function, the system predicts your next requirement based on your habits and chat content after answering your question.

Follow these steps to turn suggested questions on or off.

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
3. In the navigation pane on the left, choose **Chats**.
4. Choose **Intelligent Q&A > Suggested Questions** and click  to enable the function.

Figure 6-4 Enabling Suggested Questions



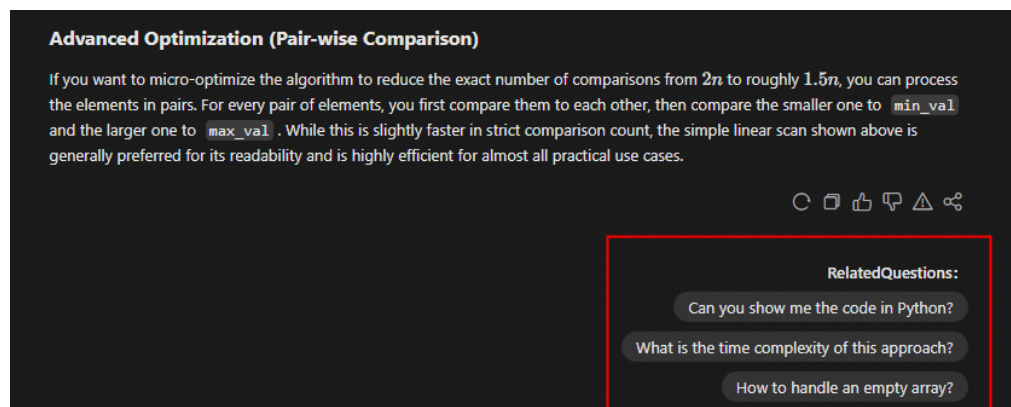
- At the bottom of the input box on the chat panel, choose **Built-in > Ask** to switch to the intelligent Q&A mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

- Once this function is enabled, it not only answers your questions but also intelligently predicts your follow-up needs based on your habits and context.

Figure 6-5 Display effect of Suggested Questions



Input Optimization

CodeArts Agent optimizes user input to help beginners ask questions more clearly and professionally, enabling users to obtain comprehensive, structured, and high-quality responses at one time. This effectively reduces repeated follow-up questions and enhances communication efficiency as well as user experience.

You can refer to the following steps to optimize your input:

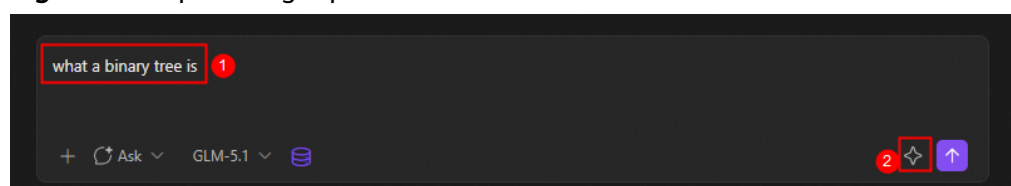
- Switch to the chat mode.**
 - Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
 - At the bottom of the input box on the chat panel, choose **Built-in > Ask** to switch to the intelligent Q&A mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

- Enter a prompt, such as "what a binary tree is," and click . Wait until the optimization is complete.

Figure 6-6 Optimizing input



If the optimization result does not meet your expectations, click  to restore to the original prompt.

Code Explanation

Allows the system to walk you through complex code with function, purpose, application scenario, and main logic. This clear explanation helps you get up to speed faster, and make knowledge flows smoothly, keeping the whole team in sync.

You can refer to the following steps to explain code:

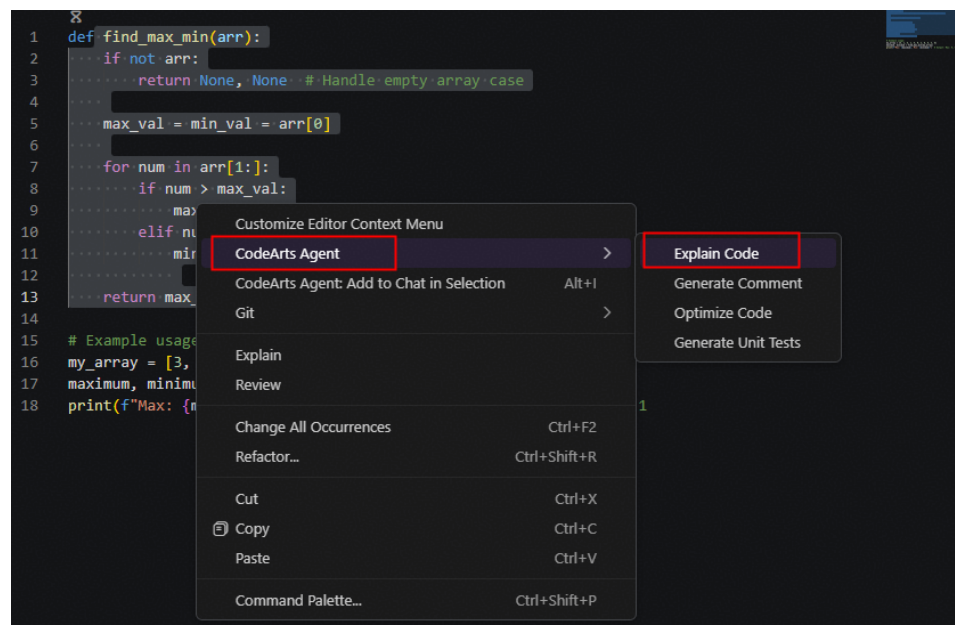
1. **Switch to the chat mode.**
 - a. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
 - b. At the bottom of the input box on the chat panel, choose **Built-in > Ask** to switch to the intelligent Q&A mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

NOTE

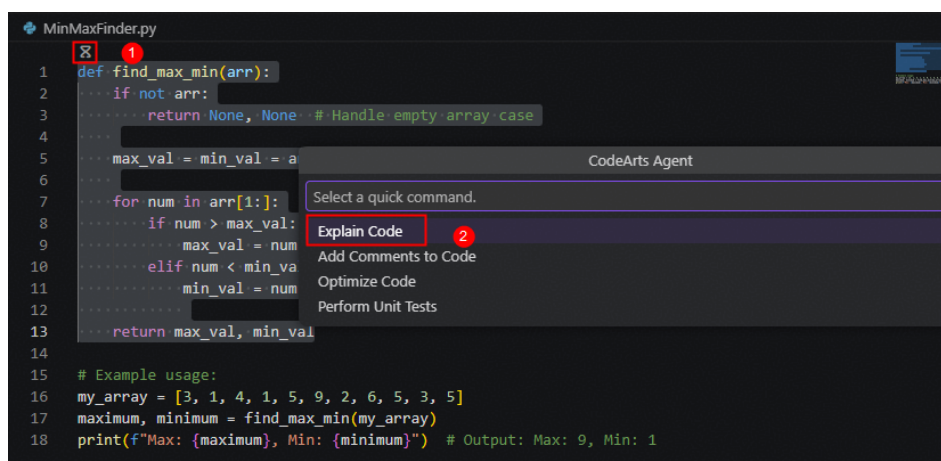
If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

2. Send a code explanation task.
 - Method 1: In the code editing area, right-click the target code and choose **CodeArts Agent > Explain Code**.

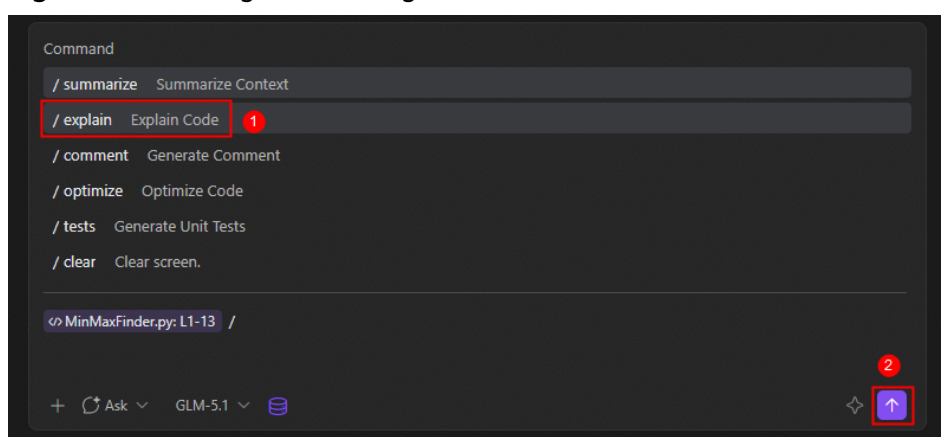
Figure 6-7 Right-clicking to send a task



- Method 2: In the code editing area, click  above the target code and choose **Explain Code** from the displayed shortcut menu.

Figure 6-8 Sending a task using a shortcut command in the editing area

- Method 3: In the code editing area, right-click the target code and choose **CodeArts Agent: Add to Chat in Selection** to add the selected code to the chat box as the context. Then in the chat box, enter `/`, or choose  > **Shortcuts**. Select `/explain` and click .

Figure 6-9 Sending a task using a shortcut command in the chat box

3. View the explanation of the code's function, purpose, and application scenarios.

Code Commenting

Allows the system to analyze the code's architecture and logic, then auto-generate clear, meaningful comments. It facilitates standardizing the comment style across your entire repository, so readability becomes second nature and code archaeology is a thing of the past.

You can refer to the following steps to add comments to the code:

1. **Switch to the chat mode.**
 - a. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
 - b. At the bottom of the input box on the chat panel, choose **Built-in > Ask** to switch to the intelligent Q&A mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

2. Send a task to add code comments.
 - Method 1: In the code editing area, right-click the target code and choose **CodeArts Agent > Generate Comment**.
 - Method 2: In the code editing area, click  above the target code and choose **Add Comments to Code** from the displayed shortcut menu.
 - Method 3: In the code editing area, right-click the target code and choose **CodeArts Agent: Add to Chat in Selection** to add the selected code to the chat box as the context. Then in the chat box, enter `/`, or choose  > **Shortcuts**. Select `/comment` and click .
3. View the code with comments added. Click  to replace the original code with the commented version.

Code Optimization

Allows CodeArts Agent to analyze code performance bottlenecks and provide optimizations, helping users improve the readability and maintainability of code.

You can refer to the following steps to optimize code:

1. **Switch to the chat mode.**
 - a. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
 - b. At the bottom of the input box on the chat panel, choose **Built-in > Ask** to switch to the intelligent Q&A mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

2. Send a code optimization task.
 - Method 1: In the code editing area, right-click the target code and choose **CodeArts Agent > Optimize Code**.
 - Method 2: In the code editing area, click  above the target code and choose **Optimize Code** from the displayed shortcut menu.
 - Method 3: In the code editing area, right-click the target code and choose **CodeArts Agent: Add to Chat in Selection** to add the selected code to the chat box as the context. Then in the chat box, enter `/`, or choose  > **Shortcuts**. Select `/optimize` and click .
3. View the code optimization suggestions and the optimized code. Click  to replace the original code with the optimized code.

Unit Test

Unit test allows the system to automatically generate high-quality test cases based on a single method and class, safeguarding code quality and effectively reducing defect risks.

You can refer to the following steps to perform a unit test:

1. **Switch to the chat mode.**

- a. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- b. At the bottom of the input box on the chat panel, choose **Built-in > Ask** to switch to the intelligent Q&A mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

2. Send a unit test task.

- Method 1: In the code editing area, right-click the target code and choose **CodeArts Agent > Generate Unit Tests**.
- Method 2: In the code editing area, click  above the target code and choose **Perform Unit Tests** from the displayed shortcut menu.
- Method 3: In the code editing area, right-click the target code and choose **CodeArts Agent: Add to Chat in Selection** to add the selected code to the chat box as the context. Then in the chat box, enter **/**, or choose  > **Shortcuts**. Select **/tests** and click .

3. Review the generated test code and its explanation. Click to save the test code as a file.

Context

Context can help the agent accurately understand the task background and requirement boundaries. Multi-dimensional information of files, folders, Git records, and terminals is introduced to build a complete task context cognition system. It can effectively improve the accuracy of code generation and troubleshooting, and avoid deviations caused by information missing. For details, see [Context](#).

7 Scheduled Tasks

You can create and manage scheduled tasks using natural language, including periodic tasks (such as daily code checks) and one-time tasks (such as commit reminders for next day). This allows for flexible task scheduling and control.

Constraints

Table 7-1 Constraints

Category	Description
Maximum number of concurrent tasks	A maximum of three tasks are allowed for a single project.
Scheduled task time range	The execution time range of a scheduled task cannot exceed seven days (including the start day).

Creating a Scheduled Task

- Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2 On the chat panel, click  to go to the **Scheduled Task** page.
- Step 3 On the **Scheduled Task** tab, click **Create Scheduled Task**.
- Step 4 Set the scheduled task parameters and click **OK**.

Table 7-2 Scheduled task parameters

Parameter	Description
Name	Name of the scheduled task, which helps identify and manage the task.

Parameter	Description
Project Folder	Project to which the scheduled task belongs. This parameter cannot be modified.
Prompt	Enter an instruction for creating the scheduled task, for example, "Sort out the sales data of this month and generate a report."
Frequency	How often the task will be executed. • Daily: Executes the task once a day at a specified time. For example, "Back up data every day at 8:00 a.m." • Weekly: Executes the task once a week on a specified day and time. For example, "Send the weekly report every Monday at 9:00 a.m." • Cron expression: Executes the task at a time defined using the standard Cron syntax. • Periodic: Executes a task at a specified interval, for example, once every hour.

Step 5 Return to the **Scheduled Task** page. The created scheduled task is displayed in the task list.

----End

Managing Scheduled Tasks

Table 7-3 Managing scheduled tasks

Operation	Procedure
Enabling or disabling a scheduled task	On the Scheduled Task tab, click  or  next to a scheduled task to enable or disable it. A maximum of three scheduled tasks can be enabled for a single project.
Executing a scheduled task	On the Scheduled Task tab, click  to execute the scheduled task immediately. Before executing a scheduled task, ensure that it has been enabled.
Modifying a scheduled task	On the Scheduled Task tab, click  and choose Edit to modify a scheduled task.

Operation	Procedure
Deleting a scheduled task	On the Scheduled Task tab, hover over  and choose Delete to delete a scheduled task immediately. The scheduled task is removed from the list, indicating that it is successfully deleted. CAUTION Deleted scheduled tasks cannot be restored. Exercise caution when performing this operation.
Checking the execution records of a scheduled task	On the Execution Records tab, set the start time and end time to check the execution details of all scheduled tasks in this period. The execution status can be Completed, Pending, or Running.

8 Context

8.1 Adding Context

CodeArts Agent supports incorporating context to improve Q&A accuracy. Context helps agent understand the background and intent of your question better to generate more practical and valuable answers.

A variety of context types are currently supported, including files, folders, symbols, Git changes, terminal output, problems, and knowledge bases. All contexts can be flexibly combined and seamlessly associated with your input to build richer and more accurate Q&A scenarios.

Constraints

Table 8-1 Constraints

Category	Description
Quantity	• The (custom) agent supports up to 20 contexts at a time. • The intelligent chat supports up to 20 contexts at a time.
Size	The context size must not exceed four times the context window of the currently used model. For example, if DeepSeek-V3.2 (which supports a context window of 160K tokens) is used and your context is in a file, the file size should be kept within 640K (160K × 4) tokens. For details about the context length supported by each model, see Table 8-2.

Adding Context

Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

Step 2 At the bottom of the input box on the chat panel, choose **Built-in > Ask** to switch to the intelligent Q&A mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

Step 3 Add a context.

- Method 1: Enter **#** in the input box and select the required context type from the displayed menu.
- Method 2: Click  in the input box, choose **Add Context**, and select the required context from the displayed menu.

Step 4 Context usage example

For example, when you open a project and need to fix a bug in the login function, CodeArts Agent can help you quickly locate and rectify the fault based on the context.

- File: **src/auth/login.js file** (View the login logic.)
- Folder: **src/components** (View the existing structure.)
- Git: View the latest related commits and learn the change history.
- Problems: View error logs and learn about specific error information.
- Terminal: View the test output and details about the test failure.
- Rules: View security specifications for bug fixing.

----End

Supported Context Types

CodeArts Agent supports multiple context types, which can be flexibly combined. Your input can also be combined with the context to form a complete requirement expression. Multiple context types are supported, including files, folders, symbols, Git changes, terminal outputs, problems, and knowledge bases.

File

You can ask questions about one or more files or edit them.

Procedure:

In the chat window of CodeArts Agent, enter **#** in the input box, or choose  > **Add Context**. Click **File** and select the file you want from the list. To add multiple files, repeat the preceding operations.

Folder

You can query and modify all content in a specified folder. When you need to perform batch operations on a code module, you only need to select the corresponding project directory to complete related tasks.

Procedure:

In the chat window of CodeArts Agent, enter **#** in the input box, or choose  **>** **Add Context**. Click **Folder** and select the folder you want from the list. To add multiple folders, repeat the preceding operations.

Symbol

You can reference code symbols such as functions, classes, and variables as the context.

Procedure:

In the chat window of CodeArts Agent, enter **#** in the input box, or choose  **>** **Add Context**. Click **Symbol** and select the function, class, or variable to be referenced.

NOTE

When CodeArts Agent is switched to the (custom) agent mode, symbols cannot be added as contexts.

Git

You can analyze code changes committed to the code repository, helping developers easily view each code change and clearly understand the project evolution and quality trend.

Procedure:

In the chat window of CodeArts Agent, enter **#** in the input box, or choose  **>** **Add Context**. Click **Git** and select the commit code or checkpoint to be referenced from the list.

Terminal

You can quickly locate historical commands and execution results through the terminal.

Procedure:

In the chat window of CodeArts Agent, enter **#** in the input box, or choose  **>** **Add Context**. Click **Terminal** and select **TerminalFull** (complete command history) or **TerminalLastCommand** (last terminal command) as required.

Problems

You can use **Problems** to quickly reference compilation error information, analyze all problems, and provide corresponding solutions.

Procedure:

In the chat window of CodeArts Agent, enter **#** in the input box, or choose  **>** **Add Context**. Click **Problems** and ask questions on the error information.

Knowledge Bases

You can use knowledge bases to ask questions about the overall knowledge system, improving the capabilities of code generation and R&D Q&A scenarios.

Before using this function, you need to create a knowledge base. For details, see [Knowledge Bases](#).

Procedure:

In the chat window of CodeArts Agent, enter # in the input box, or choose  > **Add Context**. Click **Knowledge Bases** and select the required knowledge base and knowledge set.

Rules

You can customize rules to help CodeArts Agent better understand the code repository structure and comply with project specifications. These rules can be automatically loaded or invoked by CodeArts Agent, or manually applied by you. Create rules before using them. For details, see [Rules](#).

To manually apply a rule, perform the following operations:

In the chat window of CodeArts Agent, enter # in the input box, or choose  > **Add Context**. Click **Rules** and select the rule you want from the list. To add multiple rules, repeat the preceding operations.

Using the Editor Content as the Context

When you edit code files in CodeArts Agent, it can view the active file by default. You can directly ask CodeArts Agent questions related to this file, such as asking about a specific block of code within it.

In the code editor, select the corresponding code. Right-click and choose **CodeArts Agent: Add to Chat in Selection** to add the selected content to the chat box as the context.

Context Compression

CodeArts Agent provides a context window (as shown in [Figure 8-1](#)) to display the token usage in the current chat, helping you view the context usage and avoid affecting the chat continuity.

When your development proceeds around a topic, the chat content accumulates and the context window is about to overflow. You can compress the current context. CodeArts Agent automatically summarizes the context, reducing the usage cost and improving the response efficiency.

NOTE

When you want to discuss a new topic or start a new task, you are advised to start a new chat. This can effectively avoid interference from irrelevant historical records, so that CodeArts Agent can more accurately understand and respond to your requirements.

Figure 8-1 Context window

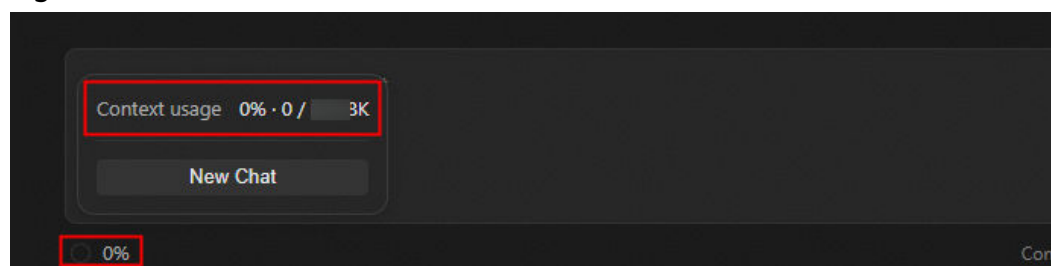


Table 8-2 Context compression rules

Mode	Model	Context Limit	Context Compression Mode
Intelligent Q&A	GLM-5.1	198.0K tokens	- Manual compression: If context usage goes over 10% of the limit, click Summarize. - Automatic compression: When the total token count of the current and historical chats exceeds 90% of the limit, the context is automatically compressed.
Agent Custom agent	GLM-5.1 GLM-5.2	198.0K tokens	Automatic compression: When the total token count of the current and historical chats exceeds 90% of the limit, the context is automatically compressed.

8.2 Rules

Constraints are imposed on agent behavior through custom rules to enable a more accurate understanding of and strict compliance with relevant specifications.

Table 8-3 Comparison between rules and other functions

Dimension	Rule	Skill	MCP Server
Definition	Constrains the behavior of an agent.	Describes how to complete a specific task.	Invokes external tools.
Loading method	Is loaded into the context at the beginning of a chat for continuous inference.	Is loaded on demand to reduce context usage.	Is not involved in the inference process; calls external interfaces as required.
Application scenario	Defines code specifications, output formats, team collaboration agreements, etc. Example: Code must comply with PEP 8 specifications.	Encapsulates testing processes, development tasks, complex business logic, etc. Example: executing UI automated testing.	Connects to external systems to perform specific operations. Example: controlling browser-related operations.

Dimension	Rule	Skill	MCP Server
Key differences	Controls how an agent should behave, defining the behavior boundary.	Specifies how to complete a task, serving as a process guide.	Provides the capability to invoke tools , representing the execution dimension.

Rule Types

Rules are classified into **local** and **cloud** agents by creation method, and are divided into **enterprise-level**, **team-level**, **project-level**, and **user-level** agents by scope of application. You can select the most appropriate rules according to specific development scenarios and collaboration needs, achieving flexible and efficient code management.

Table 8-4 Rule types

Creation Method	Scope of Application	Description
Local	Project	Effective only for the current project, distributed along with the repository, and stored locally. Rule file storage directory: ./.codeartsdoer/rule in the root directory of the project.
	User	Effective for all projects of the current user, formulated based on personal habits or preferences, and stored locally. Local storage directory: %USERPROFILE%/.codeartsdoer/rule
Cloud	Enterprise	Rules unified across the enterprise, available to all members, created by the enterprise administrator, and stored in the cloud.
	Team	Rules unified across the team, available to specific teams, created by the enterprise administrator or team administrator, and stored in the cloud.
	User	Rules formulated based on personal habits or preferences, available only to the members themselves, and stored in the cloud.

If rules with the same name exist, the rule invocation priority is as follows: **cloud enterprise rules > cloud team rules > local project rules > local user rules > cloud user rules.**

Constraints

Only the **basic** and **professional** packages support the creation of cloud rules.

Creating Local Rules

CodeArts Agent client allows the creation of both **project** and **user** rules. They are stored locally. You can select rules based on development scenarios and collaboration requirements to achieve efficient and accurate code management.

Creating Local Project Rules

Custom rules for the project are shared alongside the code repository, helping teams work consistently and efficiently under common guidelines.

- Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
- Step 3** In the navigation pane on the left, choose **Skills and Rules**. The **Installed** tab is displayed.
- Step 4** On the **Project** tab, click  next to a rule.
- Step 5** Set parameters for creating a local project rule as instructed in [Table 8-5](#).

Figure 8-2 Creating a project rule

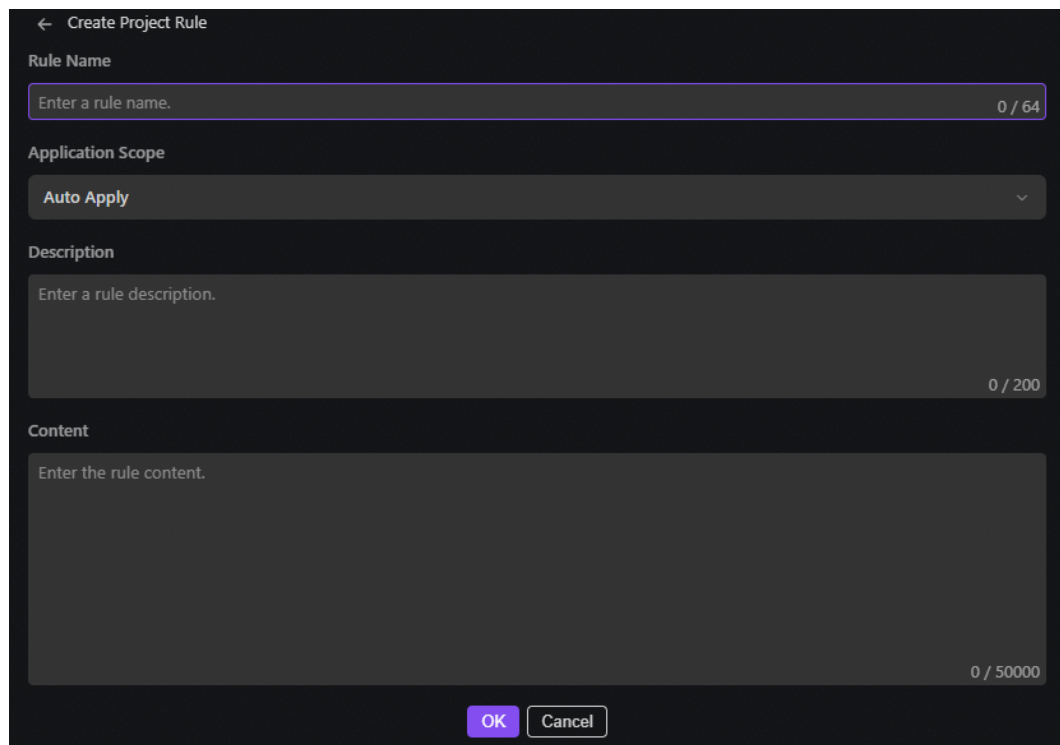


Table 8-5 Parameters for creating a local project rule

Parameter	Description
Rule Name	Name of the rule to be created. Naming rules: Enter 1 to 64 characters. The following special characters are not allowed: \/:*?"<>
Applicable Scope	Application scope of a rule, that is, where the rule takes effect. • Auto Apply: The rule applies to all chats by default and is automatically identified by the system. • Manual Call: The rule does not take effect automatically. You must enter # in the chat area and select a specific rule from the list to apply it.
Description	(Optional) Description of the rule. Enter up to 200 characters.
Content	Rule content, instructing the agent to respond consistently and predictably in certain situations. Enter up to 50,000 characters.

Step 6 Click **OK** to complete the rule creation.

On the **Project** tab, view the created local project rule. It is stored in the `./codeartsdoer/rule` directory of the project root directory.

----End

Creating Local User Rules

User rules are created according to personal habits and preferences. CodeArts Agent applies these rules consistently across all projects to deliver results tailored to user style.

Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

Step 2 Click  in the upper right corner of CodeArts Agent IDE to open the settings page.

Step 3 In the navigation pane on the left, choose **Skills and Rules**. The **Installed** tab is displayed.

Step 4 On the **User** tab, click  next to a rule.

Step 5 Set parameters for creating a local user rule as instructed in [Table 8-6](#).

Figure 8-3 Creating a local user rule

← Create User Rule

Rule Name

Enter a rule name.0 / 64

Application Scope

Auto Apply

Type

Local

Description

Enter a rule description.0 / 200

Content

Enter the rule content.0 / 50000

OK

Cancel

Table 8-6 Parameters for creating a local user rule

Parameter	Description
Rule Name	Name of the rule to be created. Naming rules: Enter 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
Applicable Scope	Application scope of a rule, that is, where the rule takes effect. ● Auto Apply: The rule applies to all chats by default and is automatically identified by the system. ● Manual Call: The rule does not take effect automatically. You must enter # in the chat area and select a specific rule from the list to apply it.
Type	Location where the rule is stored. If you select Local , the user rule is stored locally.
Description	(Optional) Description of the rule. Enter up to 200 characters.
Content	Rule content, instructing the agent to respond consistently and predictably in certain situations. Enter up to 50,000 characters.

Step 6 Click **OK** to complete the rule creation.

On the **User** tab, view the created local user rule. It is stored in the **%USERPROFILE %/.codeartsdoer/rule** directory on the local.

----End

Defining Rules by Creating an AGENTS.md File

CodeArts Agent rules are now compatible with the **AGENTS.md** file. You only need to place the **AGENTS.md** file into the corresponding directory, and CodeArts Agent will automatically recognize and apply the rules defined in the file.

Table 8-7 Levels of the AGENTS.md file

Level	Scope	Priority Rule	Description
User	All projects under the current account	• Existing rules take priority: When the content in AGENTS.md conflicts with existing rules, the existing rules shall prevail.	User-exclusive AGENTS.md , custom configurations based on personal habits and preferences. Local storage directory: %USERPROFILE %/.codeartsdoer/
Project	Current project	• Project configuration takes priority: When user and project AGENTS.md files coexist and their content conflicts, the project AGENTS.md shall prevail.	Project-specific AGENTS.md applies only to the current project. Storage directory: the root directory of the project

The following describes how to create a project **AGENTS.md** to customize rules and their invocation effects.

Step 1 Create a project **AGENTS.md**.

Create an **AGENTS.md** file in the root directory of the project for instructing project agent behaviors. Use either of the following methods:

- **Local creation:** Locate the target project folder on the local machine. Create a file named **AGENTS.md** in the root directory of the project.
- **Creation on CodeArts Agent:** Log in to CodeArts Agent IDE as instructed in [Quick Start](#). In the explorer area, locate the target project. Create a file named **AGENTS.md** under this project.

Example configuration of **AGENTS.md**:

```
# Project Agent Code of Conduct
## 1. Scope Declaration
This rule takes effect only for chats within the current project context.
## 2. Core Response Constraints
- Structured expression: Responses must adopt a clear hierarchical structure (such as lists, headings, and bold text) to avoid large blocks of unformatted text. The logic needs to progress step by step, with key points highlighted.
- Factual rigor: Fabricating uncertain information is strictly prohibited. For professional domain content, accuracy must be ensured. If information is missing, explicitly state "unable to confirm" instead of making guesses.
- Tone and style: Keep the tone professional yet friendly, and appropriately use relaxed and natural expressions.
## 3. Mandatory Requirements
After each response ends, the following fixed closing statement must and can only be added, and no other content shall be modified, omitted, or added:
The answer is complete. If you have any other questions, feel free to let me know at any time.
```

Step 2 Initiate a chat in the agent to verify the output effect.

The output of the agent strictly complies with the style requirements defined in **Core Response Constraints**. In addition, the standard closing statement "The answer is complete. If you have any other questions, feel free to let me know at any time." is automatically appended to the end of the session. This indicates that the custom rules in **AGENTS.md** have been successfully loaded and taken effect.

----End

Creating Cloud Rules

The cloud (CodeArts Agent console) supports the creation of three categories of rules: **enterprise rules**, **team rules**, and **user rules**, among which **cloud user rules** can be created either on the cloud or locally.



Before creating a cloud rule, ensure that you have purchased **basic** or **professional edition**.

Creating Rules on the Cloud

To create an enterprise rule, a team rule, or user rule on the cloud, perform the following steps:

Step 1 Log in to the CodeArts Agent console.

Step 2 In the navigation pane on the left, choose **Agent Settings > Skills and Rules**. The skill and rule lists are displayed.

Step 3 On the **Rules** tab, click **Add Rule** in the upper right corner. The **Add Rule** dialog box is displayed.

Step 4 Set the parameters based on [Table 8-8](#).

Figure 8-4 Creating a cloud rule

Add Rule

Name

Please enter

0/64

Content

Please enter

0/50,000

Application Scope

Always

Applied To

Private

Team

Enterprise

Teams

Please select

Tag

Please select

Brief Description

Please enter

0/1,024

Cancel

Confirm and release

OK

Table 8-8 Cloud rule parameters

Parameter	Description
Name	Name of the rule to be created. Naming rules: Enter 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
Content	Customize one or more rules. The value can contain a maximum of 50,000 characters. Example: requiring the system to always respond using Python.

Parameter	Description
Application Scope	Application scope of a rule, that is, where the rule takes effect. • Auto Apply: The rule applies to all chats by default and is automatically identified by the system. • Manual Call: The rule does not take effect automatically. You must enter # in the chat area and select a specific rule from the list to apply it.
Applied To	• Private: available only to the members themselves and will not be shared with other users. • Enterprise: available to all members of the entire enterprise, and can only be created by enterprise administrators. • Team: available to specified teams and can be created by enterprise administrators or team administrators.
Teams	This parameter is displayed only when Applied To is set to Team . Select a team from the drop-down list. If no team has been created, create one first. Enterprise administrators can view and select all teams, while team administrators can only view and select the team they belong to.
Tag	(Optional) Several types of tags have been preset in CodeArts Agent. Select tags from the drop-down list as required.
Brief Description	(Optional) Description of the rule. Enter up to 1,024 characters.

Step 5 Click **Confirm and Enable** to complete the rule creation.

On the **Rules** tab, view the created rule. To immediately view the created cloud rule on the client, go to the corresponding rule tab (**Enterprise/Team/User**) under **Skills and Rules** on the client, and click  in the rules area. Then the cloud rule is synchronized to the client.

 NOTE

- The CodeArts Agent client automatically synchronizes cloud rules every 24 hours.
- The CodeArts Agent client synchronizes only **enabled rules**.

----End

Creating a Cloud User Rule Locally

The account has been assigned a seat. If not, cloud user rules cannot be created.

Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

- Step 2 Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
- Step 3 In the navigation pane on the left, choose **Skills and Rules**. The **Installed** tab is displayed.
- Step 4 On the **User** tab, click  next to a rule.
- Step 5 Set parameters for creating a user rule as instructed in [Figure 8-5](#).

Figure 8-5 User rule

← Create User Rule

Rule Name

Enter a rule name.0 / 64

Application Scope

Auto Apply

Type

Local

Description

Enter a rule description.0 / 200

Content

Enter the rule content.0 / 50000

OK

Cancel

Table 8-9 Parameters for creating a user rule

Parameter	Description
Rule Name	Name of the rule to be created. Naming rules: Enter 1 to 64 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are allowed.
Applicable Scope	Application scope of a rule, that is, where the rule takes effect. Auto Apply: The rule applies to all chats by default and is automatically identified by the system. Manual Call: The rule does not take effect automatically. You must enter # in the chat area and select a specific rule from the list to apply it.

Parameter	Description
Type	Location where the rule is stored. If you select Cloud , the user rule is stored on the cloud.
Description	(Optional) Description of the rule. Enter up to 200 characters.
Content	Rule content, instructing the agent to respond consistently and predictably in certain situations. Enter up to 50,000 characters.

Step 6 Click **OK** to complete the rule creation.

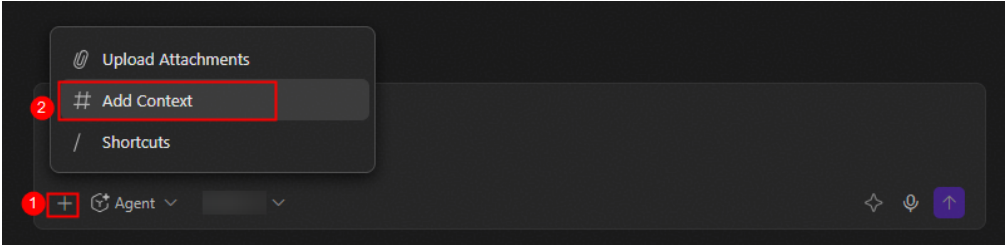
On the **User** tab, view the created cloud user rule.

----End

Referencing a Rule in Chat

- Automatically-applied rules: They take effect for all chats by default and are automatically identified by the system.
- Manually-applied rules: To use them, you need to enter # in the chat area, or click  and choose **Add Context**. Then click **Rules** and select the target rule.

Figure 8-6 Adding context to select rules



Managing Rules

CodeArts Agent IDE supports full lifecycle management of rules, helping users efficiently control and continuously optimize them based on business needs.

Managing Local Rules

Table 8-10 Managing Local Rules

Operation	Procedure
Editing a local rule	On the settings page of CodeArts Agent IDE, choose Skills and Rules > Project > Rules or Skills and Rules > User > Rules . Hover the cursor on the target rule and click  to edit the rule.

Operation	Procedure
Deleting a local rule	On the settings page of CodeArts Agent IDE, choose Skills and Rules > Project > Rules or Skills and Rules > User > Rules. Hover the cursor on the target rule and click  to delete the rule. The target rule disappears from the list once successfully deleted. CAUTION Deleting a rule will affect the agents using it. Deleted rules cannot be recovered. Exercise caution when performing this operation.

Managing Cloud Rules

Table 8-11 shows the operation permissions of different roles on **managing cloud rules**.

Table 8-11 Operation permissions of different roles on cloud rules

Role	Enterprise Rule	Team Rule	User Rule (Operations Except Editing)	User Rule (Editing)
Enterprise administrators	All operations on all rules.	All operations on all rules.	All operations except editing on all rules.	Editing on rules created by themselves.
Team administrators	The following operations can be performed: • Favorite or unfavorite rules. • Modify the application scope of cloud rules.	All operations on rules created by members of their own team (including the team administrator).	All operations except editing on rules created by members of their own team (including the team administrator).	Editing on rules created by themselves.

Role	Enterprise Rule	Team Rule	User Rule (Operations Except Editing)	User Rule (Editing)
Team members	The following operations can be performed: • Favorite or unfavorite rules. • Modify the application scope of cloud rules.	The following operations can be performed on rules created by members of their own team (including the team administrator): • Favorite or unfavorite rules. • Modify the application scope of cloud rules.	All operations except editing on rules created by themselves.	Editing on rules created by themselves.
Enterprise members	The following operations can be performed: • Favorite or unfavorite rules. • Modify the application scope of cloud rules.	No operation permissions.	All operations except editing on rules created by themselves.	Editing on rules created by themselves.

Table 8-12 Managing cloud rules

Operation	Procedure
Editing a cloud rule	1. On the CodeArts Agent console, choose Skills & Rules, click the Rules tab, and view the status of the target rule. • If it is enabled, disable it first. • If it is disabled, go to the next step to edit the rule. 2. Click Edit next to the target rule.

Operation	Procedure
Deleting a cloud rule	- On the CodeArts Agent console, choose Skills & Rules, click the Rules tab, and view the status of the target rule. - If it is enabled, disable it first. - If it is disabled, go to the next step to delete the rule. - Click Delete next to the target rule. The target rule disappears from the list once successfully deleted. CAUTION Deleting a rule will affect the agents using it. Deleted rules cannot be recovered. Exercise caution when performing this operation.
Enabling or disabling a cloud rule	On the CodeArts Agent console, choose Skills & Rules , click the Rules tab, and click  /  next to the target rule. Only enabled rules can be synchronized to the CodeArts Agent client.
Adding a cloud rule to favorites or removing it	On the CodeArts Agent console, choose Skills & Rules , click the Rules tab, and click Favorite or Unfavorite next to the target rule. View the favorites on the Favorites tab.
Batch deleting cloud rules	- On the CodeArts Agent console, choose Skills & Rules, click the Rules tab, and view the status of the target rule. - If it is enabled, disable it first. - If it is disabled, go to the next step to delete the rule. - Select the target rules and choose More > Batch Delete in the upper right corner. A confirmation dialog box is displayed. - Enter DELETE or click Autofill, and then click OK. The target rule disappears from the list once successfully deleted.
Batch enabling or disabling cloud rules	On the CodeArts Agent console, choose Skills & Rules , and click the Rules tab. Select the target rules and click Batch Enable or choose More > Batch Disable in the upper right corner. Only enabled rules can be synchronized to the CodeArts Agent client.
Batch adding cloud rules to favorites or removing them	On the CodeArts Agent console, choose Skills & Rules , and click the Rules tab. Select the target rules and choose More > Batch Favorite or More > Batch Unfavorite in the upper right corner. View the favorites on the Favorites tab.

Operation	Procedure
Modifying the application scope of cloud enterprise rules or team rules	On the settings page of CodeArts Agent IDE, choose Skills and Rules > Enterprise > Rules or Skills and Rules > Team > Rules. Locate the target rule and click the switch next to it. : You need to manually apply the rule. : The system automatically identifies the rule and applies it.
Batch modifying the application scope of cloud enterprise rules or team rules	- On the settings page of CodeArts Agent IDE, choose Skills and Rules > Enterprise > Rules or Skills and Rules > Team > Rules. Click  in the upper right corner. - Select the target rules and click Enable or Disable in the upper right corner.

Related Documents

For an end-to-end usage example of the rules, refer to AI Drawing Logic Construction Practice Based on Custom Rules. This practice configures rule constraints over the AI process of graphing $\sin(x)$ to achieve standardized generation with precise coordinates, consistent formatting, and controllable results.

8.3 Knowledge Bases

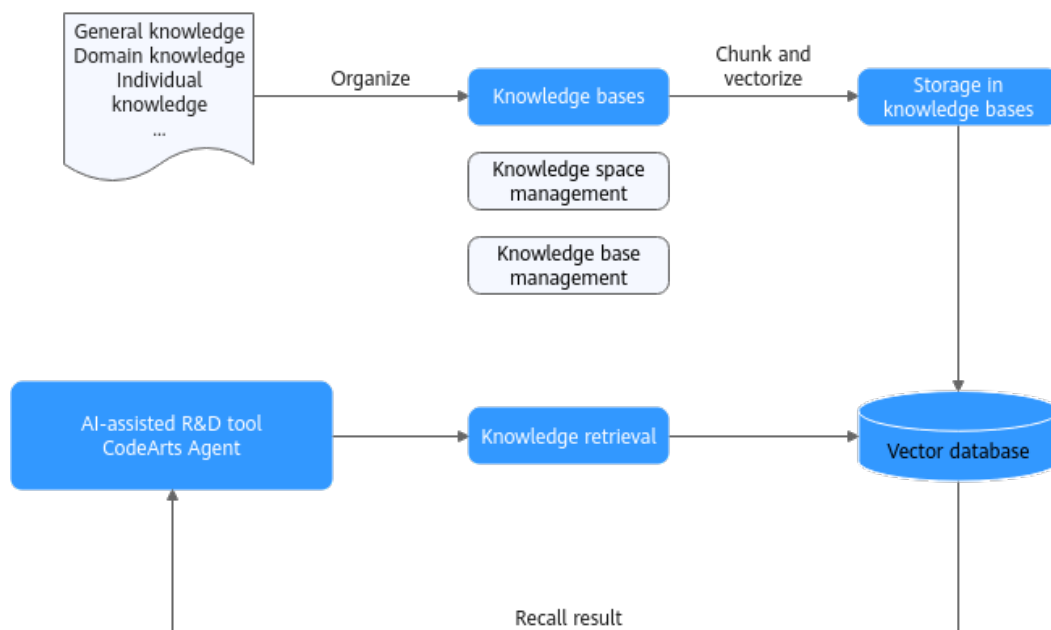
CodeArts Agent improves chat features using knowledge bases. Developers can add internal documents and technical resources to these bases for smarter responses. With company-specific information, AI provides precise answers tailored to real-world needs, boosting both development speed and decision quality.

Getting Started with Knowledge Bases

CodeArts Agent leverages knowledge bases primarily for unit test generation, code generation, and R&D Q&A, dramatically enhancing both accuracy and speed across these critical scenarios.

How Knowledge Bases Work

The following figure shows the process of how the tool processes user queries into answers visually. It includes important stages like retrieving data and making logical conclusions, making it easy for you to grasp how knowledge bases work.

Figure 8-7 How knowledge bases work

1. To fulfill service needs, knowledge is organized into distinct types, such as general, domain-specific, or individual expertise.
2. You can create a knowledge space to manage knowledge bases.
3. To create a knowledge base, upload desired files. Then, choose a chunking method to break data source into manageable pieces. Each chunk is then transformed into a vector by an embedding model.
4. CodeArts Agent may reference knowledge bases for more accurate retrieval and generation.

Basic Concepts

Learn these key concepts before using knowledge bases for quick setup.

Table 8-13 Basic concepts

Basic Concept	Description
Knowledge space	A knowledge space is a collection of knowledge formed after chunking, vectorizing, indexing user-uploaded data, and storing it in a database. It is categorized based on actual usage dimensions. A knowledge space itself does not store content; it is used solely for the centralized management of knowledge bases.
Knowledge base	A knowledge base organizes and stores different types of documents. These documents are categorized by their type and source for clear classification, making them easier to find and understand.

Basic Concept	Description
File	Files serve as the primary carrier of knowledge. To ingest them into a knowledge base for retrieval, configure a proper chunking strategy, select an embedding model, apply index filters, and define retrieval policies.
Chunk	Chunks refer to portions of a file. They are created when loading and splitting a file, serving as fundamental units for vectorization, storage, and recall.

Constraints

File storage capacity limits vary across different versions of CodeArts Agent, as detailed below:

- Basic edition: up to 5 GB
- Professional edition: up to 20 GB

Prerequisites

- You have been assigned a seat. If not, you cannot create or manage a knowledge space.
- An enterprise administrator has joined a team, if they want to create a team knowledge base.

Accessing Knowledge Bases

- Method 1: Use CodeArts Agent IDE.
 - a. On the top menu bar of CodeArts Agent IDE, click  to go to the global settings page.
 - b. Click **Knowledge Base** to open the knowledge space page in an external browser.

If you have not logged in to the CodeArts Agent console, the login page is displayed. After login, you can access the knowledge space page.
- Method 2: Use a browser.

Enter the knowledge space address in the browser and press **Enter**. The login page is displayed. Enter the username and password to log in to it.

Creating a Knowledge Base

Step 1 Create a knowledge space.

A knowledge space is used to manage multiple knowledge bases in one place. Before creating a knowledge base, you need to create a knowledge space.

1. On the **Knowledge** page, click **Create Space** and set parameters by referring to [Table 8-14](#).

Figure 8-8 Creating a knowledge space

Create Knowledge Space

×

Name

1–18 characters: Use letters, digits, hyphens (-), or underscores (_), but do not start with an underscore.

Visible To

☒ Private

☐ Team

☐ Enterprise

Tags

Select or create a tag (max. 10 tags).

Description

Describe Knowledge Space content for faster and more accurate AI access.

0/300

Cancel

OK

Table 8-14 Parameters for creating a knowledge space

Parameter	Description
Name	Indicates the name of the knowledge space to be created. Naming rules: The name cannot start with an underscore (_) and can contain a maximum of 18 characters, including letters, digits, underscores (_), and hyphens (-).
Visible To	Controls the access permissions of members. – Private: available only to members themselves. – Team: available only to team members and created by the enterprise administrator or team administrator. – Enterprise: available to all enterprise members and created by the enterprise administrator.
Teams	This parameter is displayed only when Visible To is set to Team . Select a team from the drop-down list; only the teams which the current account belongs to are displayed.

Parameter	Description
Tag	(Optional) Indicates the tags to be added to the knowledge space for content classification and management. A maximum of 10 tags can be selected. 1. If you have not created any tags yet, follow the operations below to create them. Otherwise, proceed to the next step. Select Add from the drop-down list. In the displayed dialog box, enter a tag name containing a maximum of 32 characters and click OK . 2. Select the target tag.
Description	(Optional) Indicates the description of the knowledge space. A maximum of 300 characters are allowed. A detailed description helps CodeArts Agent accurately understand the knowledge scope, improving retrieval and matching precision.

2. Click **OK**.

In the knowledge space list, you can view the created one.

Step 2 Create a knowledge base.

Create a knowledge base within a knowledge space and specify its type and source. This ensures a clear information structure and easy retrieval.

1. Select your desired knowledge space from the list to view its detailed information.
2. Click **Create Knowledge Base** in the upper right corner.
3. Configure basic information by referring to [Table 8-15](#).

Figure 8-9 Creating a knowledge base

Create Knowledge Base ×

1 Basic Information

2 Ingestion Configuration

Name

1–32 characters: Use Chinese characters, letters, digits, hyphens (-), or underscores (_), but do n

Knowledge Type

☒ Unstructured ☐ Structured

Data Source

☒ Local file ☐ Code

Refers to text content, such as articles, reports, and books. Supported formats: TXT, DOCX, PDF, MD, CSV, XLS and XLSX.

Knowledge Description

Describe the content of this knowledge base. A detailed description helps agents search and use it more effectively.

0/300

Cancel

Next

Table 8-15 Basic parameters of a knowledge base

Parameter	Description
Name	Indicates the name of the knowledge base to be created. Naming rules: The name cannot start with an underscore (_) and can contain a maximum of 32 characters, including letters, digits, underscores (_), and hyphens (-).
Knowledge Type	Indicates the knowledge base type. You can select Unstructured or Structured . – Unstructured: Data does not have a fixed format. – Structured: Data has fixed formats, fields, and structures.

Parameter	Description
Data Source	Indicates the data source of the knowledge base. You can select Local file or Code. – When Knowledge Type is set to Unstructured, the following data sources are supported: ▪ Local file: The code base mainly contains textual content, such as articles, reports, and books. Supported formats include TXT, DOCX, PDF, MD, CSV, XLS, and XLSX. ▪ Code: Only ZIP packages containing Java, Python, C, C++, or Vue code files are supported. – When Knowledge Type is set to Structured, only local JSON files are supported.
Knowledge Description	(Optional) Indicates the description of the knowledge base content. A maximum of 300 characters are allowed. A detailed description helps guide the agent to call and utilize the knowledge base more accurately.

4. Click **Next** and complete the content import configuration by referring to [Table 8-16](#).

Table 8-16 Content import configuration parameters

Parameter	Description
Embedding Model	Select an embedding model to vectorize the uploaded files/code. The following models are available. – bge-m3: a multilingual 1024-dimensional text embedding model that supports hybrid retrieval (dense, sparse, and multi-vector representation). It supports a maximum text length of 8,192 tokens, making it ideal for complex, multilingual scenarios. – m3e-base: a general-purpose 768-dimensional embedding model for Chinese and English text, suitable for most basic semantic understanding and retrieval tasks. – bge-large-zh-v1.5: a Chinese-specific model optimized for tasks such as retrieval, classification, clustering, and semantic matching. – paraphrase-multilingual-MiniLM-L12-v2: a 364-dimensional multilingual embedding model that excels in text similarity calculations, ideal for speed-sensitive scenarios. – VESO-v1.3: a specific model deeply optimized for code, delivering outstanding performance in code vectorization and search tasks.
Splitter	This parameter is displayed when Knowledge Type is set to Unstructured and Data Source is set to Local file. The way your content is split depends on the splitter you choose. Different languages can use different splitters for optimal results. – By paragraph: Splits text by double line breaks (\n\n) or paragraph structure. – By line: Splits text by single line breaks (\n). – By sentence (for English): Splits text by sentence boundaries. – By character: Splits text into individual characters (including whitespace characters).
Chunk Length	This parameter is displayed when Knowledge Type is set to Unstructured and Data Source is set to Local file. Range: 50 to 1,200 characters. Default value: 1,000 characters.

Parameter	Description
Chunk Overlap	This parameter is displayed when Knowledge Type is set to Unstructured and Data Source is set to Local file. Overlap between sequential chunks: 0 to chunk length (default value: 0).
Child Splitter	This parameter is displayed when Knowledge Type is set to Unstructured and Data Source is set to Local file. Applied when a chunk exceeds the maximum allowed length. Currently, the following child splitters are supported: – By character: Splits text into individual characters (including whitespace characters). – By sentence (for English): Splits text by sentence boundaries. – By word (for English): Splits text by whitespace characters.
Index Fields	This parameter is displayed when Knowledge Type is set to Structured. Enter the JSON keys to be used as index fields during retrieval. Up to 20 fields are supported, separated by English commas (,).
Filter Fields	This parameter is displayed when Knowledge Type is set to Structured. Enter the JSON keys to be used as filter criteria during retrieval. Up to 2 fields are supported, separated by English commas (,).
Indexed Field	This parameter is displayed when Data Source is set to Code. Select indexed fields to specify which data fields the system needs to build indexes for in advance. During retrieval, the agent will perform keyword matching only against these indexed fields. For example, if function body is selected as an indexed field, the agent will prioritize matching keywords within the function body section of the code when a user enters a search query.
Use Sliding Window	This parameter is displayed when Data Source is set to Code. Once enabled, complex functions are segmented into chunks to improve search accuracy.

5. Click **OK**.

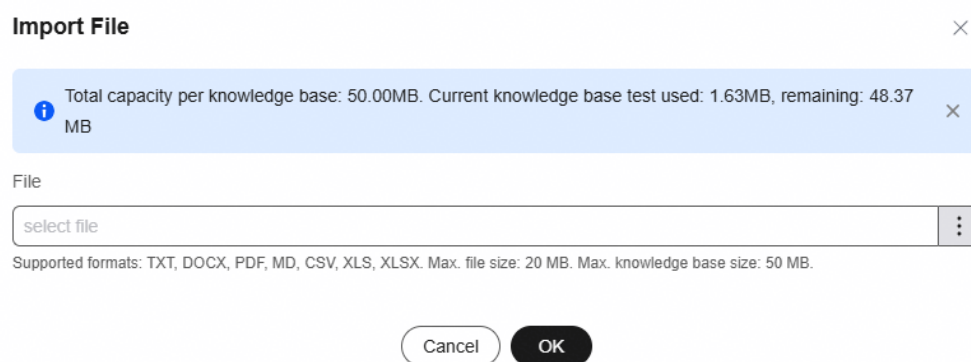
In the knowledge base list, you can view the created one.

Step 3 Import files.

After creating a knowledge base, you can upload various local documents to it. Once uploaded, the system automatically cleans the data and builds indexes, ensuring that your imported content can be quickly retrieved and called for instant knowledge accumulation.

1. Click the name of your desired knowledge base in the list to view its detailed information.
2. Click **Import File** in the upper right corner.
3. Select target code files or other files from the local and click **OK** to return to the knowledge base page.

Figure 8-10 Uploading code files or other files



You can upload code files or other files based on your required knowledge type. **The maximum size for a single file or ZIP package is 20 MB, and the maximum capacity for a single knowledge base is 50 MB.**

Table 8-17 Description for file importing

Knowledge Type	Data Source	Description
Unstructured	Local files	TXT, DOCX, PDF, MD, CSV, XLS, and XLSX files are supported.
	Code	Only ZIP packages containing Java, Python, C, C++, or Vue code files are supported.
Structured	Local files	Only JSON files are supported.

4. View the import result.

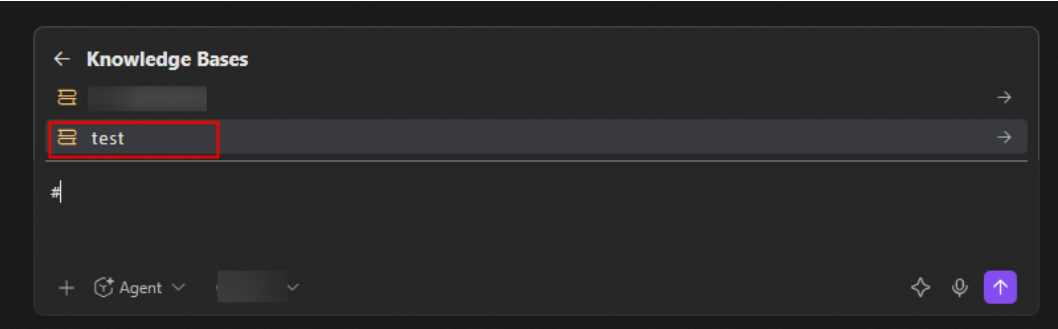
When **Ingestion Status** changes from **Uploading** to **Ingested**, the file import is successful.

----End

Referencing a Knowledge Base in the Context

- Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2 In the input box on the chat panel, enter #, or choose  > **Add Context**. Then choose **Knowledge Bases**.
- Step 3 In the knowledge base list, select the knowledge space (for example, the individual knowledge space) created in [Step 1](#).

Figure 8-11 Selecting a knowledge space



- Step 4 In the target knowledge space, select the knowledge base (for example, service introduction document) created in [Step 3](#) and add it to the context.
- Step 5 After the knowledge base is added, you can initiate a chat based on the knowledge base.

----End

Managing Knowledge Bases

[Table 8-18](#) shows the operation permissions of different roles on [managing knowledge bases](#).

Table 8-18 Operation permissions of different roles on knowledge bases

Role	Enterprise Knowledge Base	Team Knowledge Base	User Knowledge Base
Enterprise administrators	All operations on all knowledge bases.	All operations on the knowledge bases of the team which the enterprise administrator belongs to.	All operations on the knowledge bases created by themselves.

Role	Enterprise Knowledge Base	Team Knowledge Base	User Knowledge Base
Team administrators	The following operations can be performed: • Test hit rates of knowledge bases. • View operation history on knowledge bases.	All operations on knowledge bases created by members of their own team (including the team administrator).	All operations on the knowledge bases created by themselves.
Team members	The following operations can be performed: • Test hit rates of knowledge bases. • View operation history on knowledge bases.	The following operations can be performed on knowledge bases created by members of their own team (including the team administrator): • Test hit rates of knowledge bases. • View operation history on knowledge bases.	All operations on the knowledge bases created by themselves.
Enterprise members	The following operations can be performed: • Test hit rates of knowledge bases. • View operation history on knowledge bases.	No operation permissions.	All operations on the knowledge bases created by themselves.

Table 8-19 Managing knowledge bases

Operation	Procedure
Editing a knowledge space	On the knowledge space page, click  on the target knowledge space and choose Edit .

Operation	Procedure
Deleting a knowledge space	Before deleting a knowledge space, you must clear all knowledge bases in the space. 1. On the knowledge space page, click  on the target knowledge space and choose Delete. The Delete Knowledge Space dialog box is displayed. 2. Click Autofill. The system automatically enters DELETE. Click OK to delete the knowledge space. The target knowledge space has been removed from the knowledge space page, indicating that the deletion was successful. CAUTION Deleting a knowledge space will delete all resources within it. Once confirmed, data cannot be recovered; make a backup before this operation.
Testing the hit rate of a knowledge base	The hit rate test is designed to run search queries against the knowledge base to verify the accuracy and relevance of retrieval results. 1. On the knowledge space page, click Hit Test in the upper right corner. 2. Select up to 10 knowledge bases to be tested. 3. Select the field to be retrieved from the drop-down list. 4. Select a retrieval mode. • Text: Based on keyword matching; best suited for exact-word matches and short queries. • Vector: Based on semantic similarity; understanding context and synonyms, making it ideal for natural language queries. • Hybrid: Combining text and vector retrieval, using weighted fusion of both results. This generally yields better performance, especially for complex semantic queries. 5. Enter the content you want to search (maximum 1,000 characters). 6. Set the number of returned results. 7. Click Debug. You can view the final results under Chunks.

Operation	Procedure
Deleting a knowledge base	When content is severely outdated and holds no retention value, or contains critical errors where correction is too costly or impossible, follow the steps below to delete the knowledge base. 1. On the knowledge space page, click the knowledge space where the target knowledge base is located to view its details. 2. Locate the row that contains the target knowledge base, and click Delete. The Delete Knowledge Base dialog box is displayed. 3. Click Autofill. The system automatically enters DELETE. Click OK to delete the knowledge base. The target knowledge base has been removed from the knowledge space details page, indicating that the deletion was successful. CAUTION Deleting a knowledge base will delete all data within it. The deleted data cannot be recovered. Ensure that you have backed up important data before this operation.
Batch deleting knowledge bases	Deleting knowledge bases one by one manually is inefficient; you can batch delete knowledge bases through the following steps: 1. On the knowledge space page, click the knowledge space where the target knowledge base is located to view its details. 2. Click Batch Delete in the upper right corner and select the target knowledge bases. 3. Click Delete. The Batch Delete Knowledge Bases dialog box is displayed. 4. Click Autofill. The system automatically enters DELETE. Click OK to delete the knowledge bases. The target knowledge bases have been removed from the knowledge space details page, indicating that the deletion was successful. CAUTION Deleting a knowledge base will delete all data within it. The deleted data cannot be recovered. Ensure that you have backed up important data before this operation.
Viewing operation history on knowledge bases	1. On the knowledge space page, click the knowledge space where the target knowledge base is located to view its details. 2. Click History in the upper right corner to view the operation logs of all knowledge bases in the knowledge space.

Operation	Procedure
Re-importing a file	If file upload fails, check your network connection or file format, and click Reimport to try again. 1. On the knowledge space page, click the knowledge space where the target knowledge base is located to view its details. 2. Click the name of the target knowledge base to view its details. 3. Click Reimport in the upper right corner and view the import result.
Deleting a file	When dealing with a small number of files, deleting them one by one is more intuitive and flexible, making it easier to locate and clear specific content. 1. On the knowledge space page, click the knowledge space where the target knowledge base is located to view its details. 2. Click the name of the target knowledge base to view its details. 3. Click Delete next to the target file. The Delete File dialog box is displayed. 4. Click Autofill. The system automatically enters DELETE. Click OK to delete the file. The target file has been removed from the knowledge base details page, indicating that the deletion was successful. CAUTION Deleted files will be permanently cleared and cannot be recovered. Ensure that you have backed up important data before this operation.

Operation	Procedure
Batch deleting files	Deleting many files one by one manually is inefficient; you can also batch delete files using the following steps: This is ideal for scenarios such as knowledge base restructuring, version iterations, and deduplication, helping you keep your knowledge base organized. 1. On the knowledge space page, click the knowledge space where the target knowledge base is located to view its details. 2. Click the name of the target knowledge base to view its details. 3. Click Batch Delete in the upper right corner and select the target files. 4. Click Delete. The Batch Delete Files dialog box is displayed. 5. Click Autofill. The system automatically enters DELETE. Click OK to delete the files. The target files have been removed from the knowledge base details page, indicating that the deletion was successful. CAUTION Deleted files will be permanently cleared and cannot be recovered. Ensure that you have backed up important data before this operation.

8.4 Multimodal Input

CodeArts Agent supports uploading images and other attachments to integrate them into the chat context. You can directly upload or drag and drop error screenshots, code blocks, or technical documentation into the input box. This eliminates the need for lengthy text descriptions, allowing you to clearly reproduce the problem scenario and achieve precise, efficient communication of your requirements.

Constraints

Table 8-20 Constraints

Category	Description
Agent mode	• Uploading images from a local directory: You can upload 1 to 5 images at a time. Each image must not exceed 10 MB, and the total size cannot exceed 20 MB. Supported formats: PNG, JPG, JPEG, and WebP. • Uploading other attachments from a local directory: You can upload 1 to 20 items at a time with no file size limits. Supported formats: XLS, XLSX, CSV, DOC, DOCX, PPT, PPTX, PDF, MD, MDX, MDC, TXT, and SVG. • Dragging and dropping images or other files from the project directory Find the target image or file in the file tree under the explorer on CodeArts Agent IDE, and drag it into the input box to upload it.
Intelligent Q&A mode	Only local images can be uploaded. • Uploading images from a local directory: You can upload 1 to 5 images at a time. Each image must not exceed 4 MB, and the total size cannot exceed 4 MB. Supported formats: PNG, JPG, JPEG, and WebP. • Dragging and dropping images or other files from the project directory Find the target image or file in the file tree under the explorer on CodeArts Agent IDE, and drag it into the input box to upload it.
OS	The voice input function is supported only on the following OSs: • Windows OS: Windows 11 (x64) is recommended. For Windows 10 (x64), the version must be 2019 or later, and upgrading to the latest stable version is advised. • macOS: macOS 11 or later, compatible with ARM64 (Apple Silicon).

Uploading Images in a Chat

Example: uploading a code block screenshot to the agent.

Step 1 Switch to the agent mode.

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

Step 2 Upload an image.

- **Images from a local directory**

Click  in the input box, select **Upload Attachments**, and upload a local image. You can also drag and drop a local image into the input box to upload it.

- **Images from the clipboard**

Paste into the input box to upload an image from your clipboard.

- **Images from the project directory**

Find the target image in the file tree under the explorer on CodeArts Agent IDE, and drag it into the input box to upload it.

Step 3 View the image recognition results.

----End

Uploading Other Attachments in a Chat

Example: uploading a PDF file to the agent.

Step 1 Switch to the agent mode.

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

Step 2 Click  in the input box, select **Upload Attachments**, and upload a local attachment.

Step 3 Enter a prompt and click . The agent will then create and execute a task based on the prompt.

----End

Using Voice Input

Enable voice input when typing is inconvenient, when you need quick verbal communication, or when structuring text instructions is difficult. This function supports up to 10 minutes of continuous recording each time, eliminating lengthy text descriptions and helping you achieve precise and efficient communication of your needs.

When using this function for the first time, ensure that microphone access is enabled. If it is disabled, configure it in your system settings. (The exact interface may vary depending on your OS.)

- **Windows:** Go to **Start > Settings > Privacy & security** and allow microphone access.
- **macOS:** Go to **System Settings > Privacy & Security** and allow microphone access.

Step 1 Switch to the agent mode.

1. Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
2. At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

Step 2 At the bottom of the input box on the chat panel, click  and start recording. The AI will automatically convert your voice input into text and enter it in the input box.

The duration of a single voice input cannot exceed 10 minutes.

Step 3 Once you finish voice input, click  and then . The agent will automatically create a task based on your prompt.

If there are errors in the voice recognition result, you can edit the text manually before sending.

----End

8.5 Memory

Memory allows CodeArts Agent to adapt to how you work. The agent will record your habits and preferences, such as your preferred answer style and language. Once enabled, the agent will extract your personalized preferences from your historical chats and automatically apply these settings in subsequent chats, providing you with a more tailored service experience.

Enabling the memory function offers the following advantages:

- **Personalized code generation**
AI not only predicts the next line of code but also intelligently generates code based on your personal coding style. This reduces manual adjustment time, boosts development efficiency, and more accurately matches your habits and needs.
- **Contextual understanding**
When you start a new project, AI leverages its contextual understanding to recall your preferred tools and patterns (such as logging components and error handling methods), providing smarter and more efficient development support.

- **Accurate error detection**

Since AI is familiar with your coding style and habits, it intelligently identifies and alerts you to anomalies during code writing, such as missing null pointer checks or code structures that deviate from your typical style. This helps you quickly locate and fix issues.

Constraints

Table 8-21 Constraints

Category	Description
Mode	This function can be used only in agent mode.
Memorized object	• Technology stacks and professional domains: such as programming languages, frontend/backend frameworks, and styling solutions. • Coding styles and specifications: such as naming conventions and code organization methods. • Interaction and prompt habits: such as language preferences, role settings, and preferred depth of explanation.
Quantity	A maximum of 200 memory records can be displayed.
Priority	**Priority** When personal preferences conflict with skills or rules in a project, the skills and rules take precedence.

Enabling the Memory Function

Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

Step 2 Click  in the upper right corner of CodeArts Agent IDE to open the settings page.

Step 3 In the navigation pane on the left, choose **Memory**.

Step 4 Enable the memory function (enabled by default).

Step 5 At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

Step 6 Initiate a chat as you want.

For example, enter the first question.

Generate Dijkstra's algorithm in Java.

After the agent completes the response, enter the second question.

Generate binary sort in Java.

Step 7 After the agent completes the response, click  in the upper right corner of CodeArts Agent IDE to go to the global settings page.

Step 8 Choose **Memory** to view the memory records extracted by AI based on your chats.

----End

Managing Memory Records

Table 8-22 Managing memory records

Operation	Procedure
Modifying memory information	Under Settings on CodeArts Agent IDE, open the Memory page. Move the pointer to the memory information to be modified and click . CAUTION Modifying stored memory information is not recommended. After the memory information is modified, it will not be automatically updated.
Deleting memory information	1. Under Settings on CodeArts Agent IDE, open the Memory page. Move the pointer to the memory information to be deleted and click . 2. In the displayed dialog box, click OK to permanently delete the memory information. CAUTION Deleted memory information cannot be restored. Exercise caution when performing this operation.

8.6 Repository Index

CodeArts Agent provides a repository index function to help you quickly search and retrieve code-related information. This function supports creating both local user indexes and cloud-shared indexes, flexibly adapting to the requirements of different scenarios.

Constraints

Table 8-23 Constraints

Category	Description
Language	Java, JavaScript, TypeScript, and Go are supported.
Total number of code files	• Basic edition: Up to 50,000 code files can be indexed. • Professional edition: Up to 100,000 code files can be indexed.

Differences Between Local User Indexes and Cloud Indexes

Table 8-24 Differences between local user indexes and cloud indexes

Dimension	Local User Index	Cloud Index
Used by	Current user	- Enterprise: available to all enterprise members and created by the enterprise administrator, team administrator, enterprise members, or team members. - Team: available only to team members and created by the enterprise administrator, team administrator, or team members. - User: available only to members themselves.
Repository type	Unlimited	Git and GitHub repositories are supported.

Prerequisites

An enterprise administrator has joined a team, if they want to create a team repository index.

Creating Indexes

You can create local user indexes and cloud indexes.

Creating Local User Indexes

Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

Step 2 Click  in the upper right corner of CodeArts Agent IDE to open the settings page.

Step 3 In the navigation pane on the left, choose **Repository Index**.

Step 4 Enable the index function.

- Automatic indexing: After **Auto Update** is enabled, CodeArts Agent will automatically create and update indexes for repositories containing 50,000 or fewer files.

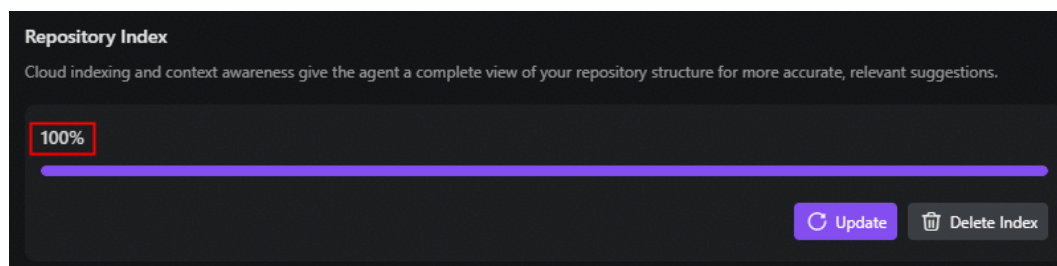
NOTE

After enabling this function, CodeArts Agent will perform incremental updates **every 5 minutes** to ensure the agent always analyzes and responds based on the latest code.

- Manual indexing: When the number of files in a local repository exceeds 50,000, you need to click **Create Index** to manually build an index.

Step 5 View the indexing results.

Once the index is successfully built, the page will show the progress at 100%.

Figure 8-12 Index created

Step 6 (Optional) Use the index in the intelligent Q&A mode.

To use the index in the intelligent Q&A mode, first ensure that the indexing process is completed. Then, at the bottom of the input box on the chat panel, click  to search for indexes. When the icon changes to , the index function is enabled.

----End

Creating Cloud Indexes

If your account has been assigned a seat, you can create cloud repository indexes. If not, you cannot create them.

Step 1 Log in to the CodeArts Agent console.

Step 2 In the navigation pane on the left, choose **Agent Settings > Repository Index**.

Step 3 Click **Create Index**.

Step 4 Set basic index information by referring to [Table 8-25](#) and click **Next**.

Table 8-25 Repository index parameters

Parameter	Description
Visible To	Visible scope of the repository index. • User: available only to members themselves. • Team: available only to team members and created by the enterprise administrator, team administrator, or team members. • Enterprise: available to all enterprise members and created by the enterprise administrator, team administrator, enterprise members, or team members.
Teams	This parameter is displayed only when Visible To is set to Team . Select a team from the drop-down list; only the teams which the current account belongs to are displayed.

Parameter	Description
Code Source	Source of the code repository that you want to index. • Git: a general Git repository. • GitHub: a GitHub repository, which can be a public repository or an authorized private repository.
Authorization Method	Method to authorize the system to access your code source. • Authorize by Service: Select a service endpoint from the drop-down list. The endpoint corresponds to the connection information of the GitHub or Git repository. If no service endpoint has been created, click Add to create one by referring to Table 8-26. • Authorize by Personal Token: Authorize access requests using an access token generated from your personal account.
Default Branch	This parameter is displayed only when Code Source is set to Git . Enter the branch name of a Git repository. The branch name can contain a maximum of 200 characters.

Figure 8-13 Creating a service endpoint

New Service Endpoint ×

code source **General Git**

Connection Name

Please enter

Git URL

For example: https://*.*.*/users/repog/git

User Name ⓘ

private-token

Password or Access Token

Git Password or Access Token 🔒

OK

Cancel

Table 8-26 Service endpoint parameters

Parameter	Description
Connection Name	Name of the custom repository to be connected. Naming rules: The name can contain letters, digits, underscores (_), and hyphens (-), but cannot start with an underscore.
Git URL	This parameter is displayed only when Code Source is set to Git . HTTPS access address of the Git repository
User Name	This parameter is displayed only when Code Source is set to Git . User name for HTTPS authentication of the Git repository.
Password or Access Token	This parameter is displayed only when Code Source is set to Git . Password or access token for HTTPS authentication of the Git repository.
Access Token	This parameter is displayed only when Code Source is set to GitHub . Access token for HTTPS authentication of the GitHub repository.

Step 5 Select the repository and language, and click **OK** to finish creating the index.

On the repository index page, view the newly created repository index. When its status changes from parsing to parsing completed, the repository index has been successfully parsed.

Step 6 Return to the CodeArts Agent IDE client and open the Git, Repo, or GitHub repository project. CodeArts Agent automatically downloads team repository indexes from the cloud.

After the team repository indexes are successfully downloaded, the progress displayed on the page reaches 100%.

 NOTE

If the number of files in the repository is greater than 50,000, click **Create Index** to manually build the repository index.

Step 7 (Optional) Use the index in the intelligent Q&A mode.

To use the index in the intelligent Q&A mode, first ensure that the indexing process is completed. Then, at the bottom of the input box on the chat panel, click  to search for indexes. When the icon changes to , the index function is enabled.

----End

Managing Repository Indexes

After a repository index is built, you can manage it throughout its lifecycle.

Managing Local Repository Indexes

Beyond creation, local repository indexes support operations such as restarting and deleting. For details, see [Table 8-27](#).

Table 8-27 Managing repository indexes

Operation	Instruction
Stop indexing	Click Stop to halt the indexing process at any time.
Restart indexing	To resume a paused indexing process, click Resume Indexing .
Update indexes	Incremental update is supported. The system compares the current status with the result of the last update, indexes the added or modified code files, and skips unchanged files to save time.
Reindex	If you need to rebuild an index entirety, click Reindex to trigger a complete indexing process.
Delete indexes	To clear all current index data, click Delete Index .
Ignore files in the current directory	If you want to exclude any file in the current folder from being indexed, click Configure . Then, in .codebaseignore , specify the file name.

Managing Repository Indexes on the Console

[Table 8-28](#) shows the operation permissions of different roles to delete repository indexes.

Table 8-28 Scope of repository indexes that can be deleted by different roles

Role	Enterprise-Level Repository Index	Team-Level Repository Index	User-Level Repository Index
Enterprise administrator	All repository indexes	All repository indexes of the administrator's team	All repository indexes created by the user
Team administrator	All repository indexes created by the user	All repository indexes created by team administrator and members of the team managed by the team administrator	All repository indexes created by the user

Role	Enterprise-Level Repository Index	Team-Level Repository Index	User-Level Repository Index
Team member	All repository indexes created by the user	All repository indexes created by the user	All repository indexes created by the user
Enterprise member	All repository indexes created by the user	No operation permissions.	All repository indexes created by the user

Table 8-29 Repository index management on the console

Operation	Procedure
Delete a specified repository index.	When there are only several indexes, you can delete them one by one. 1. On the Repository Index page, locate the target index, and click Delete in the Operation column. The Delete Repository Index dialog box is displayed. 2. Confirm the information, and click Autofill. The system automatically enters DELETE. Then, click OK. The target index cannot be found on the Repository Index page, indicating that the deletion is successful. CAUTION The deleted index cannot be restored. Exercise caution when you perform this operation.
Delete the repository indexes in a batch.	If there are many indexes, you can delete them in batches. 1. In the upper right corner of the Repository Index page, click Batch Delete and select the indexes to be deleted. 2. Click Delete. The Delete Repository Index dialog box is displayed. 3. Confirm the information, and click Autofill. The system automatically enters DELETE. Then, click OK. The target indexes cannot be found on the Repository Index page, indicating that the deletion is successful. CAUTION The deleted indexes cannot be restored. Exercise caution when you perform this operation.

9MCP

CodeArts Agent connects to an MCP server using the Model Context Protocol (MCP). It leverages the server's extra tools and resources to enhance its capabilities.

What Is MCP?

The Model Context Protocol (MCP) is an open standard that specifies how applications interact with large language models (LLMs). It creates a consistent way for LLMs to access external data and tools. The main role of the MCP server is to enable LLMs to call on external resources to help answer questions or perform tasks. For more information about MCP, see the [MCP official documentation](#).

Table 9-1 Comparison between MCP servers and other functions

Dimension	Rule	Skill	MCP Server
Definition	Constrains the behavior of an agent.	Describes how to complete a specific task.	Invokes external tools.
Loading method	Is loaded into the context at the beginning of a chat for continuous inference.	Is loaded on demand to reduce context usage.	Is not involved in the inference process; calls external interfaces as required.
Application scenario	Defines code specifications, output formats, team collaboration agreements, etc. Example: Code must comply with PEP 8 specifications.	Encapsulates testing processes, development tasks, complex business logic, etc. Example: executing UI automated testing.	Connects to external systems to perform specific operations. Example: controlling browser-related operations.

Dimension	Rule	Skill	MCP Server
Key differences	Controls how an agent should behave, defining the behavior boundary.	Specifies how to complete a task, serving as a process guide.	Provides the capability to invoke tools , representing the execution dimension.

Disclaimer

MCP servers are built, published, and maintained by third-party developers or organizations. CodeArts Agent does not review, verify, or endorse the code implementation, security, stability, data processing practices, or service quality of third-party MCP servers.

CodeArts Agent shall not be held liable for any risks arising from the use of third-party MCP servers, including but not limited to issues caused by defects in the MCP servers themselves, configuration errors, service anomalies, API changes, call failures, or the accuracy, completeness, and security of returned data.

Certain MCP servers may be inaccessible in specific countries or regions due to applicable laws and regulations, network constraints, policy adjustments by third-party services, or changes in service availability. CodeArts Agent has no control over these factors and cannot guarantee that all MCP servers will remain continuously available or fully functional across different regions or network environments.

Before using a third-party MCP server, users should independently assess its suitability and security risks, and ensure compliance with local laws and regulations as well as the terms of use and licensing requirements of the relevant third party.

For MCP servers involving sensitive operations on code, data, files, or account permissions, users are advised to fully understand their scope of permissions, data processing methods, and security policies before use, and to implement necessary security measures based on their specific business scenarios.

Configuring MCP Servers and Using Them

Currently, two types of MCP servers are supported, depending on the communication mode between the MCP server and the MCP client (CodeArts Agent).

Type	Transmission Protocol	Environment
MCP server based on standard input and output (stdio)	stdio	Local
HTTP-based MCP server	SSE	Local/Remote

Type	Transmission Protocol	Environment
	Streamable HTTP	Local/Remote

Installing MCP Servers from the MCP Marketplace

The MCP marketplace integrates featured MCP servers from the AgentArts platform, allowing you to browse and install them.

Step 1 Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

Step 2 Click  in the upper right corner of CodeArts Agent IDE to open the settings page.

Step 3 In the navigation pane on the left, choose **MCP**.

Step 4 On the **Marketplace** tab, find the required MCP server (such as Master of Ink Draft) and click  on the right to install it.

NOTE

There is no hard limit on the number of MCP servers you can add on CodeArts Agent. However, for the best performance, we recommend the following:

- Keep the total number of created servers under 20.
- Keep the number of enabled servers under 8.
- Enable 3 servers for the optimal experience.

After the installation is completed, you can view the installed MCP server on the **Installed** tab.

Step 5 Use the installed MCP server.

1. At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

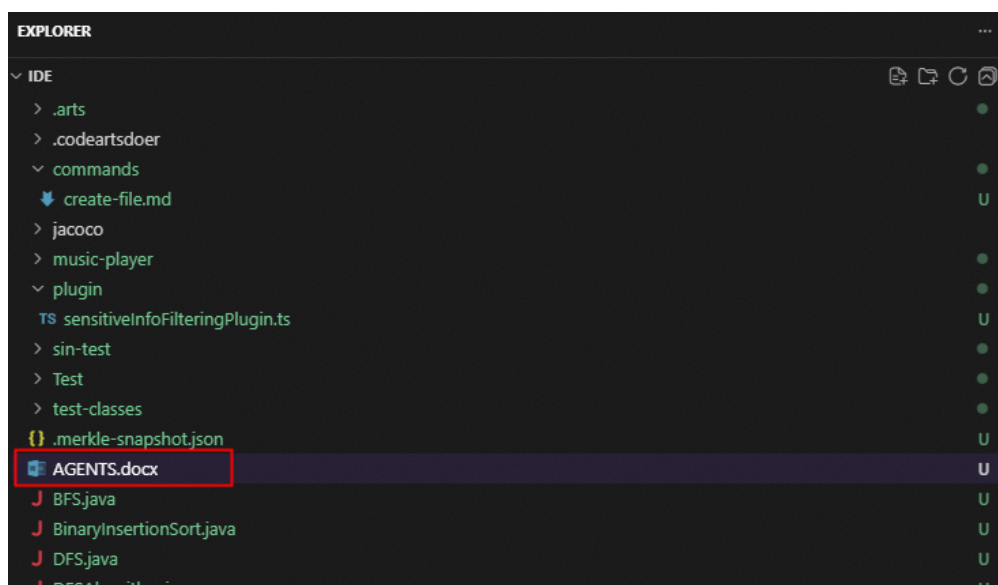
NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

2. In the input box, enter the following information and click :
Convert the AGENTS.md file to a Word file.

3. View the invocation result of the MCP tool.

If a corresponding Word file is generated in the project directory under the explorer, the MCP tool has been successfully invoked.

Figure 9-1 Viewing the invocation result

----End

Manually Creating Stdio-based MCP Servers

- Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
- Step 3** In the navigation pane on the left, choose **MCP**.
- Step 4** Click **Configure MCP** to open the `mcp_settings.json` file, and edit it.

NOTE

There is no hard limit on the number of MCP servers you can add on CodeArts Agent. However, for the best performance, we recommend the following:

- Keep the total number of created servers under 20.
- Keep the number of enabled servers under 8.
- Enable 3 servers for the optimal experience.

The following is a general template for MCP configuration:

```
{
  "mcpServers": {
    "Server name": {
      "command": "Command for starting the process",
      "args": [
        "Parameter array"
      ],
      "env": {
        "Environment variables"
      }
    }
  }
}
```

Table 9-2 Parameters

Parameter	Description
Server name	A unique name or identifier (maximum 64 characters) used to distinguish between different MCP servers. NOTE If the server name exceeds 64 characters, the request may fail with the following error. For details, refer to the definition of Tools under Core Server Features . - type: session.error info - length over limit, maximum is 64
command	(Mandatory) Enter the path of the command or program needed to start the MCP server.
args	(Optional) Enter the list of parameters for the command. Use this along with command to configure the MCP server's operation.
env	(Optional) Enter the environment variables passed to the MCP server. These variables must be strings.

For example, you want to use **Playwright** to directly control the browser to perform automatic operations such as web page navigation, click, input, screenshot, and data extraction. The **mcp_settings.json** file is configured as follows:

```
{
  "mcpServers": {
    "playwright": {
      "command": "npx",
      "args": [
        "@playwright/mcp@latest"
      ]
    }
  }
}
```

In this example, you need to install the **node.js** environment on the local host. The **node-v24.x** version is recommended. The agent can install it for you upon request.

Step 5 After the configuration is complete, press shortcut keys (**Ctrl+S** for Windows/Linux and **Command (⌘)+S** for macOS) to save the settings and wait until the tool is started.

If issues arise during setup, go back to the chat page for assistance from the agent.

Step 6 Use the installed MCP server.

- At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

2. In the input box, enter the following information and click :
Access <https://codearts.huaweicloud.com> and view all download options.
The agent invokes the **Playwright** MCP server to open the webpage. Then you can download required items.

----End

Manually Creating HTTP-based MCP Servers

- Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
- Step 3** In the navigation pane on the left, choose **MCP**.
- Step 4** Click **Configure MCP** to open the `mcp_settings.json` file, and edit it.

NOTE

There is no hard limit on the number of MCP servers you can add on CodeArts Agent. However, for the best performance, we recommend the following:

- Keep the total number of created servers under 20.
- Keep the number of enabled servers under 8.
- Enable 3 servers for the optimal experience.
- General MCP configuration template for the server-sent events (SSE) protocol; applicable to event-driven, real-time push scenarios.

```
{
  "mcpServers": {
    "Server name": {
      "url": "https://example.com/mcp/sse",
      "type": "sse"
    }
  }
}
```

- General MCP configuration template for the Streamable HTTP protocol; applicable to general streaming data transmission scenarios.

```
{
  "mcpServers": {
    "Server name": {
      "url": "https://example.com/mcp",
      "type": "streamableHttp"
    }
  }
}
```

Table 9-3 Parameters

Parameter	Description
Server name	A unique name or identifier (maximum 64 characters) used to distinguish between different MCP servers. NOTE If the server name exceeds 64 characters, the request may fail with the following error. For details, refer to the definition of Tools under Core Server Features . - type: session.error info - length over limit, maximum is 64
url	Enter the address for accessing the remote MCP server. Make sure it is a valid HTTP or HTTPS URL.
type	Communication protocol type of the MCP server. The value can be sse or streamableHttp . CAUTION If the type parameter is missing from the following example, manually add it before use to ensure normal operation in CodeArts Agent.

For example, if you want to integrate the Amap MCP server with the Amap app, configure the **mcp_settings.json** file as follows:

```
{
  "mcpServers": {
    "amap-maps-streamableHTTP": {
      "url": "https://mcp.amap.com/mcp?key=Key you have applied for on the Amap official website",
      "type": "streamableHttp"
    }
  }
}
```

Step 5 After the configuration is complete, press shortcut keys (**Ctrl+S** for Windows/Linux and **Command (⌘)+S** for macOS) to save the settings and wait until the tool is started.

If issues arise during setup, go back to the chat page for assistance from the agent.

Step 6 Use the installed MCP server.

- At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 NOTE

- If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.
- In the input box, enter the following information and click :
Help me plan a one-day tour in Beijing that includes the Palace Museum, the Great Wall, and local delicacies.

The agent will call Amap via the MCP server to generate a personalized one-day itinerary for you.

----End

Managing MCP Servers

- Enabling or disabling an MCP server: After an MCP server is installed, on the settings page of CodeArts Agent IDE, choose **MCP > Installed**. Click  next to the target MCP server to enable or disable it.
- Editing an MCP server: On the settings page of CodeArts Agent IDE, choose **MCP > Installed**. Click **Configure MCP** in the upper right corner to open the **mcp_settings.json** file.

Assume that you have installed the Master of Ink Draft MCP server from the marketplace and now need to add a server based on the Streamable HTTP protocol. The following is a configuration example of **mcp_settings.json**:

```
{
  "mcp": {
    "aigroup-mdtoward-mcp": {
      "command": "npx",
      "args": [
        "-y",
        "aigroup-mdtoward-mcp"
      ],
      "env": {},
      "description": "",
      "disabled": false,
      "timeout": 60
    },
    "amap-maps-streamableHTTP": {
      "url": "https://mcp.amap.com/mcp?key=7047c506c***526ceee",
      "type": "streamableHttp"
    }
  }
}
```

- Restarting an MCP server: On the settings page of CodeArts Agent IDE, choose **MCP > Installed**. Click  next to the target MCP server. Generally, you need to restart an MCP server in the following scenarios:
 - Applying configuration changes: If you have edited **mcp_settings.json** (such as adding new environment variables, changing startup commands, etc.), the IDE's internal cache may not update automatically. In this case, you must restart the MCP server for the new configuration to take effect immediately.
 - Service recovery: When the MCP server becomes unresponsive, freezes, or encounters errors, restarting it can reset the process state.
- Deleting an MCP server: On the settings page of CodeArts Agent IDE, choose **MCP > Installed**. Click  next to the target MCP server.

CAUTION

To minimize memory consumption and avoid cluttering your context space, promptly delete or disable any MCP servers you do not frequently use.

10 Skills

Skills are capability encapsulation mechanisms that organize professional knowledge, such as coding experience and solutions to common problems, into reusable modules. They are folders holding commands, scripts, and resources. These specialized packages can be loaded by the agent when needed to enhance task performance and efficiency.

Table 10-1 Comparison between skills and other functions

Dimension	Rule	Skill	MCP Server
Definition	Constrains the behavior of an agent.	Describes how to complete a specific task.	Invokes external tools.
Loading method	Is loaded into the context at the beginning of a chat for continuous inference.	Is loaded on demand to reduce context usage.	Is not involved in the inference process; calls external interfaces as required.
Application scenario	Defines code specifications, output formats, team collaboration agreements, etc. Example: Code must comply with PEP 8 specifications.	Encapsulates testing processes, development tasks, complex business logic, etc. Example: executing UI automated testing.	Connects to external systems to perform specific operations. Example: controlling browser-related operations.
Key differences	Controls how an agent should behave, defining the behavior boundary.	Specifies how to complete a task, serving as a process guide.	Provides the capability to invoke tools , representing the execution dimension.

Skill Directory Structure

A skill is a folder with a necessary **SKILL.md** file and optional resources. It bundles the knowledge and tools needed to perform particular tasks.

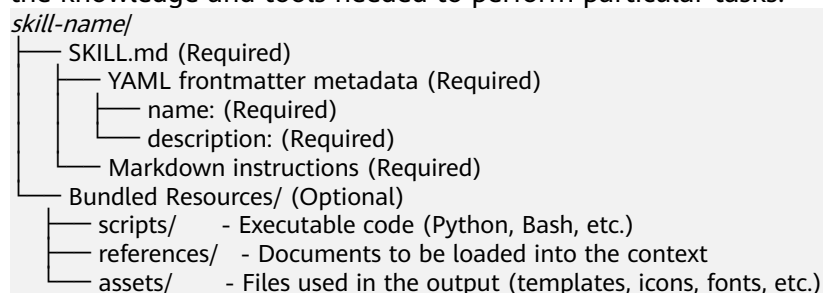


Table 10-2 Skill parameters

Parameter	Description
skill-name	Serves as the root directory for the entire skill. Replace <i>skill-name</i> with your specific skill's name, such as data-analysis-skill . The skill name consists of lowercase letters, digits, and hyphens (-). It cannot start or end with a hyphen, cannot contain consecutive hyphens, and must be 1 to 64 characters in length.
SKILL.md	Serves as the core description file of a skill. Its name cannot be modified. It includes the following essential sections: ● YAML frontmatter metadata: Placed at the very top of the SKILL.md file and enclosed by --- at the beginning and end, this section provides structured information readable by AI. It contains two fields: name and description. – name: The skill name consists of lowercase letters, digits, and hyphens (-). It cannot start or end with a hyphen, cannot contain consecutive hyphens, and must be 1 to 64 characters in length. The value of name must be the same as that of skill-name. – description: The skill description contains a maximum of 1,024 characters. This field is critical for CodeArts Agent to recognize the skill's usage scenarios, and must clearly describe the skill functions and trigger conditions. ● Markdown instructions: The Markdown content below the YAML metadata, used to detail the usage methods, parameters, examples, and dependencies of the skill.

Parameter	Description
Bundled Resources (optional)	Holds files needed for the skill. While not mandatory, it enhances the skill's functionality. • scripts/: contains executable files like Python and shell scripts that perform specific tasks for the skill. • references/: holds essential documents like APIs, domain knowledge, and company policies that support agent skills. It acts as the agent's knowledge base. Documents are loaded into the context when needed for skill execution. • assets/: contains resource files like fonts, company logos, and PowerPoint templates. The agent uses these files directly in its output without loading them into the context window.

Writing SKILL.md

Here is an example of a **SKILL.md** file that includes **YAML frontmatter** and **Markdown** content:

```
---
name: safe-file-reader
description: Read files without making changes. Use when you need read-only files access.
---

# Safe File Reader

This Skill provides read-only file access.

## Instructions

1. Use Read to view file contents
2. Use Grep to search within files
3. Use Glob to find files by pattern
```

Adding Support Files

Create other files in the same directory as **SKILL.md**, for example:

```
my-skill/
├── SKILL.md
├── reference.md
├── examples.md
├── scripts/
│   └── helper.py
├── templates/
│   └── template.txt
```

In **SKILL.md**, specify the support files to be referenced, for example:

For advanced usage, see reference.md.

Run the helper script:

```
```bash
```

```
python scripts/helper.py input.txt
```

## Skill Types

Skills are categorized into three main types: **built-in system agents**, **built-in marketplace agents**, and **custom agents**. These correspond to out-of-the-box underlying capabilities, verified industry expertise, and flexibly customizable extension capabilities, fully meeting task processing needs of agents in various scenarios.

**Table 10-3** Skill types

Type	Description
System built-in	Built-in basic skills in the system, which are globally available and ready to use out of the box. No extra configuration is required to enjoy stable and continuous basic functional support. <b>Disabling built-in skills will limit the agent's ability to handle related tasks.</b>
Built-in marketplace agents	Built-in expert skills in the system, covering core domains such as unit testing, code review, document generation, security inspection, compilation/building, and code refactoring. All skills have undergone automated testing and specification review, aiming to provide developers with professional and reliable capability support to boost development efficiency.
Custom agents	Customized skills that are independently developed by or imported from users based on business requirements. For details about the specific types of custom skills, see <a href="#">Table 10-4</a> .

## Classification of Custom Skills

Skills are classified into **local** and **cloud** skills by creation method, and are divided into **enterprise-level**, **team-level**, **project-level**, and **user-level** skills by scope of application. You can select the most appropriate skills according to specific development scenarios and collaboration needs, achieving flexible and efficient code management.

**Table 10-4** Classification of custom skills

Creation Method	Scope of Application	Description
Local	Project	Applied to the current project and stored in a local path Storage directory: <code>./.codeartsdoer/skills/</code> in the root directory of the project
	User	Effective for all projects of the current user, and stored locally Local storage directory: <code>%USERPROFILE%\.codeartsdoer/skills</code>
Cloud	Enterprise	Skills unified across the enterprise, available to all members, created by the enterprise administrator, and stored in the cloud.
	Team	Skills unified across the team, available to specific teams, created by the enterprise administrator or team administrator, and stored in the cloud.
	User	Available only to the user and stored in the cloud.

If skills with the same name exist, the skill invocation priority is as follows: cloud enterprise skills > cloud team skills > local project skills > local user skills > cloud user skills > built-in system skills.

## Constraints

Only the **basic** and **professional** packages support the creation of cloud skills.

## Creating Local Skills

CodeArts Agent client allows the creation of both **project** and **user** skills. They are stored locally. In addition, you can quickly integrate skills by **importing existing files** or **installing them from the marketplace**. These flexible and efficient methods help you easily extend development capabilities.

## Installing Skills Using npx Commands

CodeArts Agent allows the installation of **project** or **user** skills using npx commands.

**Step 1** Refer to the [official documentation](#) to install Node.js and npm.

After the installation is complete, run the following command in the terminal to verify that Node.js and npm are successfully installed:

```
npx -v
```

If the version number is displayed in the command output, Node.js and npm have been successfully installed.



**Step 5** View the installed skill.

- Go to the local directory **%USERPROFILE%/.codeartsdoer/skills** to find the target skill vercel-react-best-practices.
- On the settings page of CodeArts Agent IDE, choose **Skills and Rules**. On the **User** tab, the target skill vercel-react-best-practices is displayed and is enabled.

**Step 6** Use the skill.

In the chat window of CodeArts Agent, enter the following information and click



Use the skill vercel-react-best-practices to refactor code.

After the skill is loaded, a success message is displayed. AI will use this skill to refactor the React and Next.js code under this project.

----End

## Adding Skills from the Marketplace

Multiple out-of-the-box skills are available in the marketplace to help you quickly boost development efficiency. The following describes how to find and add these skills.

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).**Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.**Step 3** In the navigation pane on the left, choose **Skills and Rules**. The **Installed** tab is displayed.**Step 4** Click the **Marketplace** tab and search for skills.

**Table 10-5** Searching for skills

Search Method	Procedure
Source	You can obtain <b>official</b> and <b>featured</b> skills in the marketplace.      • <b>Official</b>: skills provided by CodeArts Agent.    • <b>Featured</b>: screened third-party skills.      Select a skill source from the <b>All Sources</b> drop-down list on the top of the page. You can also enter a skill name or keyword in the search box to quickly find skills.
Name	Click the sorting icon next to <b>By name</b> in the upper right corner of the page to find skills by name.
Update time	Click the sorting icon next to <b>Last updated</b> in the upper right corner of the page to find skills by update time.

**Step 5** Add skills.

1. Use either of the following methods to display the dialog box for selecting the skill installation location:
  - Click  next to the target skill.
  - Click the name of the target skill to go to its details page, and then click **Create** in the upper right corner.

**If a created or imported skill shares the same name as a skill in the marketplace, that marketplace skill will be displayed as installed.**

2. Select the installation location for the skill as a project or user skill and click **OK**.

After the installation is successful,  next to the skill changes to **Installed**.

**Step 6** (Optional) Update skills, change skill installation locations, or uninstall skills based on your actual needs.

**Table 10-6** Managing skills

Operation	Procedure
Updating a skill	<b>If a created or imported skill shares the same name as a skill in the marketplace but has a different version, that marketplace skill will be displayed as installed and to be updated. Performing an update will overwrite the original skill content with the skill content from the marketplace.</b>   Update a skill using either of the following methods:      • Move the cursor to the skill to be updated and click . After the update is successful, the <b>To be updated</b> label in the upper right corner of the skill automatically disappears.    • Click the name of the skill to be updated to access the skill details page. Click <b>Update</b> in the upper right corner. After the update is successful, the <b>To be updated</b> label in the upper right corner of the skill automatically disappears.
Modifying the skill installation location	1. Click the name of the skill whose installation location is to be modified. The skill details page is displayed.    2. Click <b>Project</b> or <b>User</b> in the upper right corner. In the displayed dialog box, select the target installation location.      When the skill installation location changes from the original to the modified one, the modification is successful.

Operation	Procedure
Uninstalling a skill	Uninstall a skill using either of the following methods:      • Move the cursor to the skill to be uninstalled and click . After the uninstallation is successful,  changes to .    • Click the name of the skill to be uninstalled to access the skill details page. Click <b>Uninstall</b> in the upper right corner. After the uninstallation is successful, <b>Uninstall</b> and <b>Project/User</b> in the upper right corner change to <b>Create</b>.

----End

## Manually Creating Local Project Skills

A skill customized for the current project is distributed along with the repository. **A local project skill takes effect immediately after creation, and the agent determines when to use it.**

- Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
- Step 3** In the navigation pane on the left, choose **Skills and Rules**. The **Installed** tab is displayed.
- Step 4** On the **Project** tab, click  next to a skill to go to the page for creating a skill.
- Step 5** Set parameters for creating a local project skill as instructed in [Table 10-7](#).

**Table 10-7** Parameters for creating a local project skill

Parameter	Description
Skill Name	Name of the project skill to be created, for example, data-cleaner-skill.   Naming rules: The name consists of lowercase letters, digits, and hyphens (-). It cannot start or end with a hyphen, cannot contain consecutive hyphens, and must be 1 to 64 characters in length.
Description	Description of the skill. Enter up to 1,024 characters.
Instruction	Instructions of the skill, used to elaborate on its usage, parameters, examples, and dependencies. Enter up to 50,000 characters.

- Step 6** Click **OK** to complete the skill creation.

On the **Project** tab, view the created local project skill. After a skill is created, it is enabled by default. Its file is stored in the `./codeartsdoer/skills` directory of the project's root directory.

----End

## Manually Creating Local User Skills

Customize your skills based on your habits and preferences. CodeArts Agent uses these settings to deliver tailored results for **every project**. **Once added, local user skills apply instantly, and the agent decides when to activate them.**

- Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
- Step 3** In the navigation pane on the left, choose **Skills and Rules**. The **Installed** tab is displayed.
- Step 4** On the **User** tab, click  next to a skill to go to the page for creating a skill.
- Step 5** Set parameters for creating a local user skill as instructed in [Table 10-8](#).

**Table 10-8** Parameters for creating a local user skill

Parameter	Description
Skill Name	Name of the user skill to be created, for example, data-validator-skill.  Naming rules: The name consists of lowercase letters, digits, and hyphens (-). It cannot start or end with a hyphen, cannot contain consecutive hyphens, and must be 1 to 64 characters in length.
Description	Description of the skill. Enter up to 1,024 characters.
Instruction	Instructions of the skill, used to elaborate on its usage, parameters, examples, and dependencies. Enter up to 50,000 characters.

- Step 6** Click **OK** to complete the skill creation.

On the **User** tab, view the created local user skill. After a skill is created, it is enabled by default. Its file is stored in the local `%USERPROFILE %/.codeartsdoer/skills` directory.

----End

## Importing Skills

You can conveniently invoke and use relevant functions by importing a local skill file into your CodeArts Agent.



---

**CAUTION**

Before importing, verify that the skill file meets the standards specified in [Skill Directory Structure](#) and [Writing SKILL.md](#). For example, the root directory of each skill folder must contain a **SKILL.md** file, which must include the skill name, description, and other details in YAML format. If the skill file does not comply with these specifications, adjust it before importing, as the import may otherwise fail.

---

- Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).
- Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.
- Step 3** In the navigation pane on the left, choose **Skills and Rules**. The **Installed** tab is displayed.
- Step 4** On the **Project** or **User** tab, click  and choose **Import**.
- Step 5** In the displayed dialog box, select a local skill ZIP file and click **OK**.

---

**CAUTION**

Only one ZIP file can be uploaded at a time. The file can contain multiple skill folders, and the total file size cannot exceed 5 MB.

---

After the import is successful, the skill name is displayed in the skill list.

Example: If you need to import both skills in DOCX and PPTX files into CodeArts Agent at the same time, you can place both skill folders into a single ZIP file and then upload it to achieve a batch upload.

----End

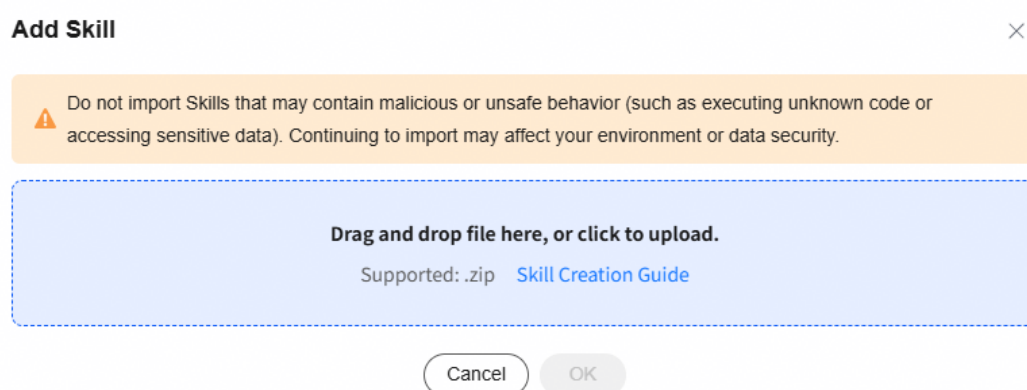
## Creating Skills on the Cloud

To create an enterprise skill, a team skill, or user skill on the cloud, perform the following steps: Before creating a cloud skill, ensure that you have purchased **CodeArts Agent basic** or **professional edition**.

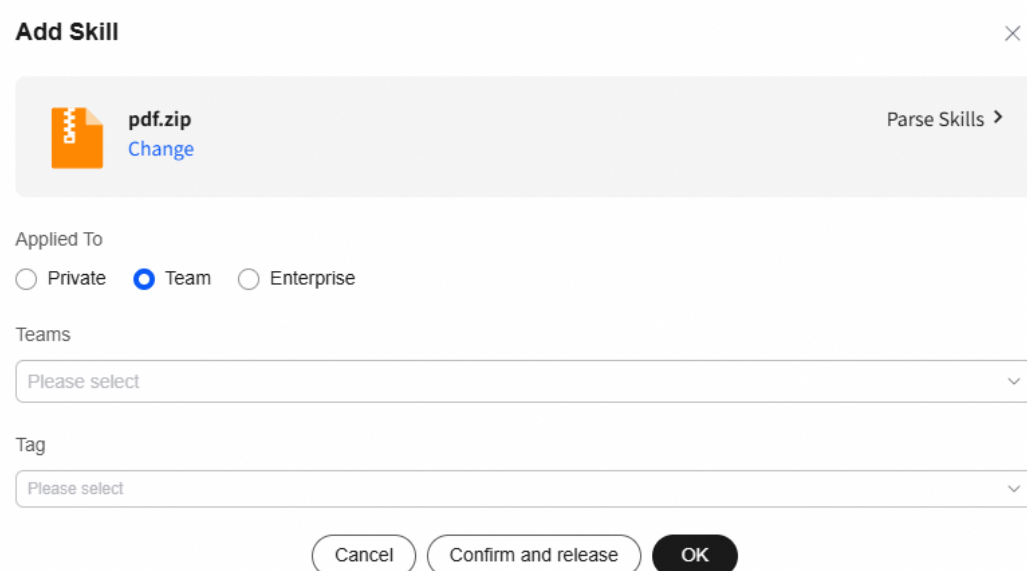
- Step 1** Log in to the CodeArts Agent console.
- Step 2** In the navigation pane on the left, choose **Agent Settings > Skills and Rules**. The skill and rule lists are displayed.
- Step 3** On the **Skills** tab, click **Add Skill** in the upper right corner. The **Add Skill** dialog box is displayed.
- Step 4** Right-click in the blue box area and upload the target skill ZIP file from the local. Alternatively, drag and drop the local skill ZIP file to the blue box area and click **OK**.

**CAUTION**

- The skill ZIP file must conform to the required directory structure; otherwise, the import may fail. For example, the root directory of each skill folder must contain a **SKILL.md** file, which must include the skill name, description, and other details in YAML format.
- Only one ZIP file can be uploaded at a time. The file can contain multiple skill folders, and the total file size cannot exceed 5 MB.

**Figure 10-3** Creating a skill

**Step 5** Set parameters for creating a cloud skill as instructed in [Table 10-9](#).

**Figure 10-4** Creating a cloud skill

**Table 10-9** Cloud skill parameters

Parameter	Description
Change	If you need to re-upload a skill ZIP file, click <b>Change</b> and re-upload it as instructed in <a href="#">Step 4</a> .
Applied To	• <b>Private</b>: available only to yourself and will not be shared with other users.    • <b>Team</b>: available to specified teams and <b>can be created by enterprise administrators or team administrators</b>.    • <b>Enterprise</b>: available to all members of the entire enterprise, <b>and can only be created by enterprise administrators</b>.
Teams	This parameter is displayed only when <b>Applied To</b> is set to <b>Team</b> . Select a team from the drop-down list. If no team has been created, create one first. Enterprise administrators can view and select all teams, while team administrators can only view and select the team they belong to.
Tag	(Optional) Several types of tags have been preset in CodeArts Agent. Select tags from the drop-down list as required.

**Step 6** Click **Confirm and Enable** to complete the skill creation.

On the **Skills** tab, view the created skill. To immediately view the created cloud skill on the client, go to the corresponding skill tab (**Enterprise/Team/User**) under **Skills and Rules** on the client, and click  or choose  > **Update** in the skills area. Then the cloud skill is synchronized to the client.

 **NOTE**

- The CodeArts Agent client automatically synchronizes cloud skills every 24 hours.
- The CodeArts Agent client synchronizes only **enabled skills**.

----End

## Using Skills

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

**Step 2** At the bottom of the input box on the chat panel, choose **Built-in > Agent** to switch to the agent mode. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 **NOTE**

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

**Step 3** Use a skill in either of the following ways:

- Method 1: Enter the skill name to invoke it.

In the input box of the chat panel, enter the following information and click



Use skill-creator to generate a set of standards that must be followed during a drawing process, named drawing-standards, with the following content:

- The horizontal axis is the x-axis.
- The vertical axis is the y-axis.
- The origin O(0,0) must be clearly marked.
- The arrows, scales, and unit lengths of the axes must be consistent.
- Tick marks should be thin solid lines.
- Axis lines should be slightly thicker solid lines.

- Method 2: Invoke a skill using the / command.

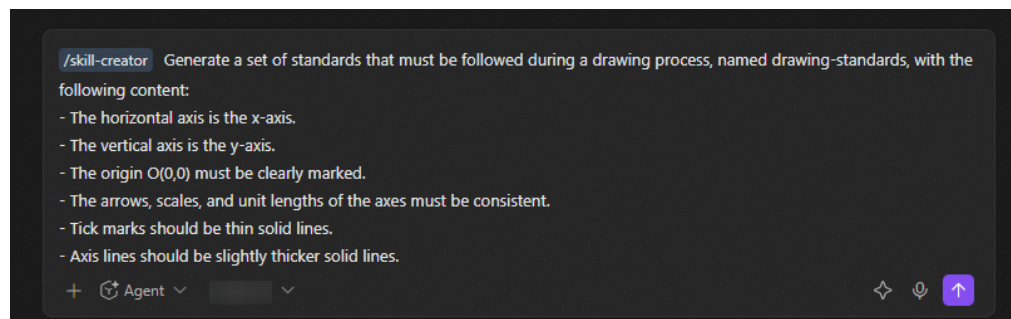
Before invoking a skill using the / command, ensure that **the skill is in the enabled state**. You can select a maximum of **20** skills at a time in the input box.

In the input box of the chat panel, enter /, select **/skill-creator** from the displayed menu, enter the following content, and click .

Generate a set of standards that must be followed during a drawing process, named drawing-standards, with the following content:

- The horizontal axis is the x-axis.
- The vertical axis is the y-axis.
- The origin O(0,0) must be clearly marked.
- The arrows, scales, and unit lengths of the axes must be consistent.
- Tick marks should be thin solid lines.
- Axis lines should be slightly thicker solid lines.

**Figure 10-5** Invoking skills using the / command

**Step 4** View the skill invoking result.

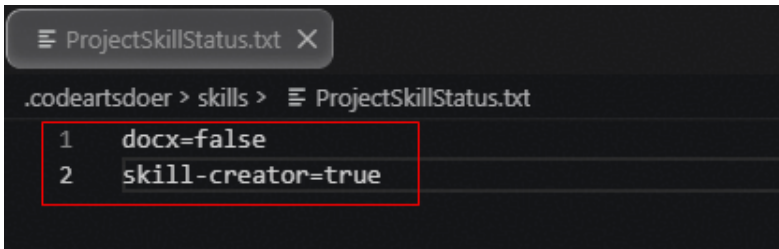
----End

## Managing Skills

CodeArts Agent IDE features comprehensive lifecycle management on skills. It allows you to finely manage and continuously optimize existing skills based on your actual business scenarios.

## Managing Local Skills

**Table 10-10** Managing local skills

Operation	Procedure
Enabling or disabling a local skill	On the settings page of CodeArts Agent IDE, choose <b>Skills and Rules &gt; Project &gt; Skills</b> or <b>Skills and Rules &gt; User &gt; Skills</b>. Toggle a target skill on or off.   When a <b>project skill</b> is created, CodeArts Agent generates a file named <b>ProjectSkillStatus.txt</b> in the <b>./codeartsdoer/skills/</b> directory at the project root to record the enablement status of each skill within the project. Each line records the status of one skill in the format of <i>Skill name=true</i> (indicating it is enabled) or <i>Skill name=false</i> (indicating it is disabled).   After a skill is deleted, its corresponding record in the <b>ProjectSkillStatus.txt</b> file is also removed to ensure that the file content stays consistent with the project status.   <b>Figure 10-6</b> Example of the ProjectSkillStatus file  
Batch enabling or disabling local skills	- On the settings page of CodeArts Agent IDE, choose <b>Skills and Rules &gt; Project &gt; Skills</b> or <b>Skills and Rules &gt; User &gt; Skills</b>. Click  in the upper right corner and choose <b>Batch Operate</b>.   - Select the target skills and click <b>Enable</b> or <b>Disable</b> in the upper right corner.
Editing a local skill	On the settings page of CodeArts Agent IDE, choose <b>Skills and Rules &gt; Project &gt; Skills</b> or <b>Skills and Rules &gt; User &gt; Skills</b>. Hover the cursor on the target skill and click  to edit the skill.
Deleting a local skill	On the settings page of CodeArts Agent IDE, choose <b>Skills and Rules &gt; Project &gt; Skills</b> or <b>Skills and Rules &gt; User &gt; Skills</b>. Hover the cursor on the target skill and click  to delete the skill.   The target skill is successfully deleted when it is removed from the skill list.   <b>CAUTION</b> Deleting a skill will affect the agents using it. Deleted skills cannot be recovered. Exercise caution when performing this operation.

Operation	Procedure
Viewing a local skill	On the settings page of CodeArts Agent IDE, choose <b>Skills and Rules</b> > <b>Project</b> > <b>Skills</b> or <b>Skills and Rules</b> > <b>User</b> > <b>Skills</b> . Hover the cursor on the target skill and click  to view the skill details.

## Managing Cloud Skills

**Table 10-11** shows the operation permissions of different roles on **managing cloud skills**.

**Table 10-11** Operation permissions of different roles on cloud skills

Role	Enterprise Skill	Team Skill	User Skill (Operations Except Editing)	User Skill (Editing)
Enterprise administrators	All operations on all skills.	All operations on all skills.	All operations except editing on all skills.	Editing on skills created by themselves.
Team administrators	The following operations can be performed:    - View cloud skill details.   - Favorite or unfavorite skills.	All operations on skills created by members of their own team.	All operations except editing on skills created by members of their own team.	Editing on skills created by themselves.
Team members	The following operations can be performed:    - View cloud skill details.   - Favorite or unfavorite skills.	The following operations can be performed on skills created by members of their own team:    - View cloud skill details.   - Favorite or unfavorite skills.	All operations except editing on skills created by themselves.	Editing on skills created by themselves.

Role	Enterprise Skill	Team Skill	User Skill (Operations Except Editing)	User Skill (Editing)
Enterprise members	The following operations can be performed:     • View cloud skill details.    • Favorite or unfavorite skills.	No operation permissions.	All operations except editing on skills created by themselves.	Editing on skills created by themselves.

**Table 10-12** Managing Cloud Skills

Operation	Procedure
Viewing cloud skill details	• On the CodeArts Agent console, choose <b>Skills &amp; Rules</b>, click the <b>Skills</b> tab, and then click the name of the target skill to view its details.    • On the settings page of CodeArts Agent IDE, choose <b>Skills and Rules</b>. Go to the corresponding skill tab (<b>Enterprise/Team/User</b>). Hover the cursor on the target skill and click  to view the skill details.
Editing cloud skills	1. On the CodeArts Agent console, choose <b>Skills &amp; Rules</b>, click the <b>Skills</b> tab, and view the status of the target skill.    • If it is enabled, disable it first.    • If it is disabled, go to the next step to edit the skill.        2. Click <b>Edit</b> next to the target skill.
Deleting a cloud skill	1. On the CodeArts Agent console, choose <b>Skills &amp; Rules</b>, click the <b>Skills</b> tab, and view the status of the target skill.    • If it is enabled, disable it first.    • If it is disabled, go to the next step to delete the skill.        2. Click <b>Delete</b> next to the target skill. The target skill is successfully deleted when it is removed from the skill list.      <b>CAUTION</b> Deleting a skill will affect the agents using it. Deleted skills cannot be recovered. Exercise caution when performing this operation.

Operation	Procedure
Enabling or disabling a cloud skill	Use either of the following methods to enable or disable a cloud skill:     - On the CodeArts Agent console, choose <b>Skills &amp; Rules</b>, click the <b>Skills</b> tab, and click  next to the target skill.   - On the settings page of CodeArts Agent IDE, choose <b>Skills and Rules</b>. Go to the corresponding skill tab (<b>Enterprise/Team/User</b>). Toggle a target skill on or off.      <b>Only enabled skills can be synchronized to the CodeArts Agent client.</b>
Adding a cloud skill to favorites or removing it	On the CodeArts Agent console, choose <b>Skills &amp; Rules</b>, click the <b>Skills</b> tab, and click <b>Favorite</b> or <b>Unfavorite</b> next to the target skill.   View the favorites on the <b>Favorites</b> tab.
Batch deleting cloud skills	- On the CodeArts Agent console, choose <b>Skills &amp; Rules</b>, click the <b>Skills</b> tab, and view the status of the target skill.   - If it is enabled, disable it first.   - If it is disabled, go to the next step to delete the skill.       - Select the target skills and choose <b>More &gt; Batch Delete</b> in the upper right corner. A confirmation dialog box is displayed.   - Enter <b>DELETE</b> or click <b>Autofill</b>, and then click <b>OK</b>. The target skill is successfully deleted when it is removed from the skill list.
Batch enabling or disabling cloud skills	Use either of the following methods to batch enable or disable cloud skills:     - On the CodeArts Agent console, choose <b>Skills &amp; Rules</b>, and click the <b>Skills</b> tab. Select the target skills and click <b>Batch Enable</b> or choose <b>More &gt; Batch Disable</b> in the upper right corner.   - On the settings page of CodeArts Agent IDE, choose <b>Skills and Rules</b>. Go to the corresponding skill tab (<b>Enterprise/Team</b>). Click  in the upper right corner. Select the target skills and click <b>Enable</b> or <b>Disable</b> in the upper right corner.   - On the settings page of CodeArts Agent IDE, choose <b>Settings &gt; Skills and Rules &gt; User</b>. Choose  &gt; <b>Batch Operate</b> in the upper right corner. Select the target skills and click <b>Enable</b> or <b>Disable</b> in the upper right corner.      <b>Only enabled skills can be synchronized to the CodeArts Agent client.</b>



Operation	Procedure
Batch adding cloud skills to favorites or removing them	On the CodeArts Agent console, choose <b>Skills &amp; Rules</b>, and click the <b>Skills</b> tab. Select the target skills and choose <b>More &gt; Batch Favorite</b> or <b>More &gt; Batch Unfavorite</b> in the upper right corner.   View the favorites on the <b>Favorites</b> tab.

# 11 Web-based Preview

---

CodeArts Agent IDE includes a built-in browser feature that lets you preview your web application's performance directly and carry out necessary actions. Specifically, you can:

- Instant external URL preview: Enter an external URL in the browser address bar to instantly preview the website.
- One-click selection import to chat: Select one or multiple elements from the browser and add them to the chat box.
- Direct element-to-agent interaction: Select any element and send it to a chat with one click to interact directly with the agent.
- Real-time console log viewing and reuse: Open the console, check the browser's printed logs, then paste them into the chat box.
- Drag-and-drop element movement: Select an element and intuitively adjust its position using the mouse.
- Two-way editing: Modify element properties directly in either the design interface or the CSS code.
- Element-level commenting: Add comments to precisely bind style optimization requirements to specific elements.

## Adjusting Page Elements in a Project

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

### NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

**Step 2** In the explorer, expand the folder, right-click the **\*\*\*.html** file, and choose **Preview in Browser**.

**Step 3** Improve selected items on the project page by adjusting their size, placement, style, or layers according to the design needs. This ensures a clean layout, logical structure, and seamless user experience.

Method 1: Add specific elements to the chat box for optimization.

1. On the displayed page, click the element selector tool .

**Figure 11-1** Selecting an element



2. Click the element to be added and choose  from the displayed menu to add the element to the input box on the chat panel.
  3. Type your prompt in the input box, like "Change the font color of selected items to blue", and press .
- Once executed, the chosen elements' font turns blue.

Method 2: Enter optimization instructions in the chat box.

1. On the displayed page, click the element selector tool .
2. Click the element to be added and choose  from the displayed menu.
3. Enter optimization keywords and press . The agent will optimize the element based on your prompts.

Method 3: Adjust the visual style in the style settings.

1. On the displayed page, click the element selector tool .
2. Click the element to be added and choose  from the displayed menu for setting the style.
3. In the style attribute panel, make adjustments as needed.

**Step 4** Enhance the element's card design. Open your browser tab, select the element picker tool , hover over the desired element, and choose **Optimize**.

**Step 5** Open your browser tab, click , and select **Improve Page** to instantly enhance how the page looks.

**Step 6** Improve the design of every file in the project. Open your browser, click , and choose **Improve Project** to quickly upgrade the appearance of all pages.

**Step 7** Use the CSS inspector to optimize the page. Open your browser, click the CSS inspector , and select the target element. You can then modify the selected element in the component tree and design panel.

In the component tree, right-click the target element to copy, cut, paste, or delete the element. You can also drag an element to another component tree node.

On the design panel, you can edit elements in a visualized manner, such as adjusting the position, layout, text, background, and border. You can also view or edit CSS style rules in the CSS.

---

 **CAUTION**

When setting the font in **Design > Text**, manually enter the name of the font installed on your local computer.

---

**Step 8** Optimize the page by adding comments. Add comments to precisely bind style optimization requirements to specific elements. The agent performs local optimization based on the comment instructions to achieve precise modifications, significantly reducing communication costs and rework rates.

1. On the displayed page, click the comment icon .
2. Select the content to which you want to add a comment, enter the optimization prompt, and click **Add Comment**.  
The system temporarily stores your comments in a bubble. You can hover the mouse pointer over the bubble to view the comment target and content. You can also click **Submit**. The agent will optimize the page based on your comments.
3. Select another content to which you want to add a comment, enter the optimization prompt, and click **Add Comment**.
4. After adding a comment, click **Submit**. The agent will optimize the page based on your comments.

To cancel saving your comments in a batch, click **Cancel** in the upper right corner. To undo the last operation, click .

---

**CAUTION**

A maximum of five comments can be added.

---

**Step 9** Preview the page on multiple devices. After a page is developed, you can preview the page on multiple devices.

1. On the displayed browser page, click  to open the device toolbar.
2. Select a device type and view the page effect.

----End

## Generating a New Page Based on an External Website

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

**Step 2** Select **Agent** at the bottom of the input box on the chat panel. The selected model is displayed on the right. You can select a model as required from the drop-down list.

 **NOTE**

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

**Step 3** Open an external website.

In the chat box, enter a prompt and click  to open the target webpage.

 **NOTE**

To view another webpage, type its URL into the browser's address bar and click  to load it instantly.

**Step 4** Generate a new page using elements from the external webpage.

1. In the input box, enter the following prompt and click .  
In the **TestDemo\_01** folder of the current project, build a search page for the browser's content. Start the preview using **node-server**. Follow all prompts and execute the given instructions as directed.
2. After the execution is complete, in the explorer, expand the **demo** folder, right-click the generated **\*\*\*.html** file, and choose **Preview in Browser**.
3. Type the website's URL into the address bar, click the element selector tool , and pick the needed element from the page.
4. In the input box, enter the following prompt and click . Once done, check the results.  
Adjust the top-searched document details to fit this project's search page layout, and preview the generated webpage.

----End

## Related Operations

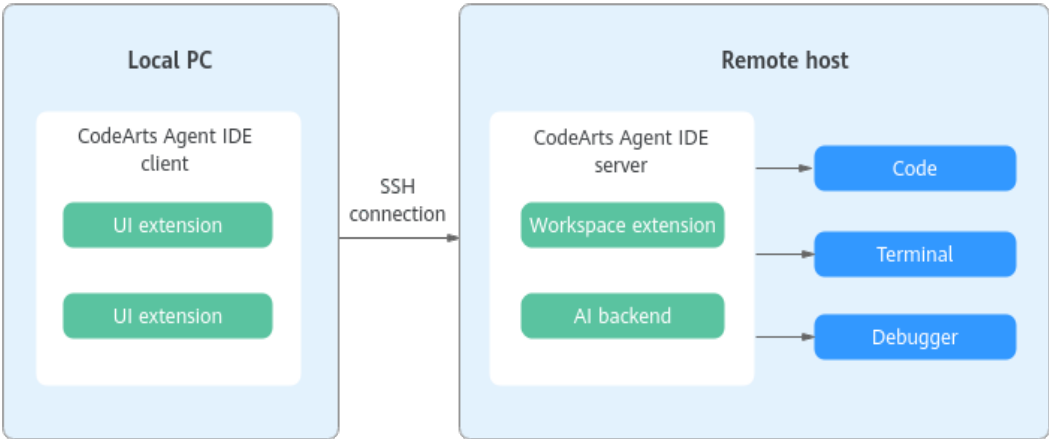
Click the developer tool icon  near the address bar in your browser. The tool shows live updates for info, warning, or error logs in your project. (You can **click Custom Levels** on the **Console** tab to view the log type.) Copy and paste these logs into CodeArts Agent to get quick analysis and solutions.

# 12 Remote Development

## 12.1 SSH Development

The Remote SSH function of CodeArts Agent IDE allows users to directly access and manage files on remote hosts from their local PC, enabling highly efficient remote development and maintenance without downloading files locally. With this function, you can utilize the complete development toolchain directly within the IDE—including code generation, debugging, and AI assistance—greatly improving the convenience and efficiency of remote work.

Figure 12-1 Remote SSH connection



### Constraints

Table 12-1 Constraints

Category	Description
Remote environment	The openssh-server has been installed in the remote target environment.

Category	Description
Configuration	The <b>AllowTcpForwarding</b> parameter in the <b>/etc/ssh/sshd_config</b> file is set to <b>yes</b> .
OS version	• <b>glibc: 2.18 or later</b> # Check the glibc version. ldd --version    • <b>glibcxx: 3.4.22 or later</b> # Check the glibcxx version. # Ubuntu strings /usr/lib/x86_64-linux-gnu/libstdc++.so.6   grep GLIBCXX # openEuler strings /lib64/libstdc++.so.6   grep GLIBCXX
Login script	The system login script (typically ~/.bashrc) should include detection logic for non-interactive logins to ensure that the script is executed only within interactive terminals. # Add the following at the beginning of the ~/.bashrc login script: # If not running interactively, don't do anything case \$- in *) ;; *) return;; esac
Webpage preview	Direct opening of the browser for preview via the file:/// protocol is currently not supported.

## Connecting to a Remote Host

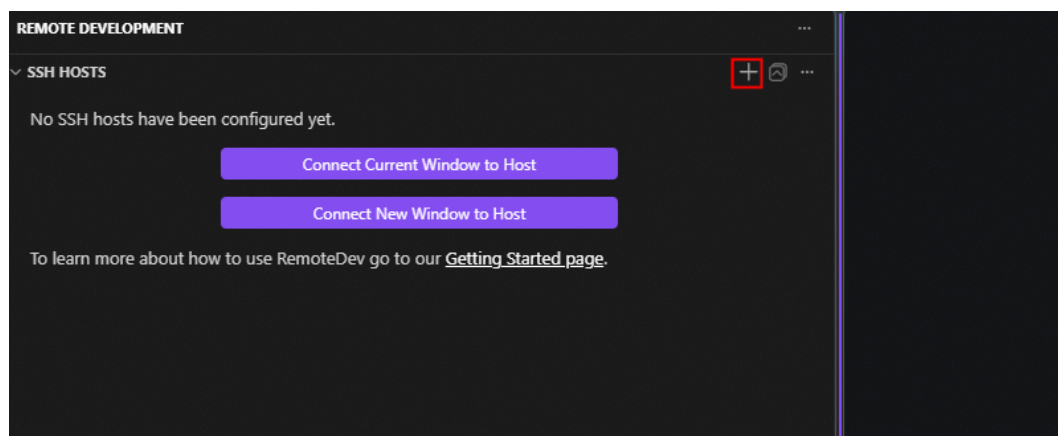
**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

**Step 2** Click  in the left menu bar of the IDE to open the remote development page.

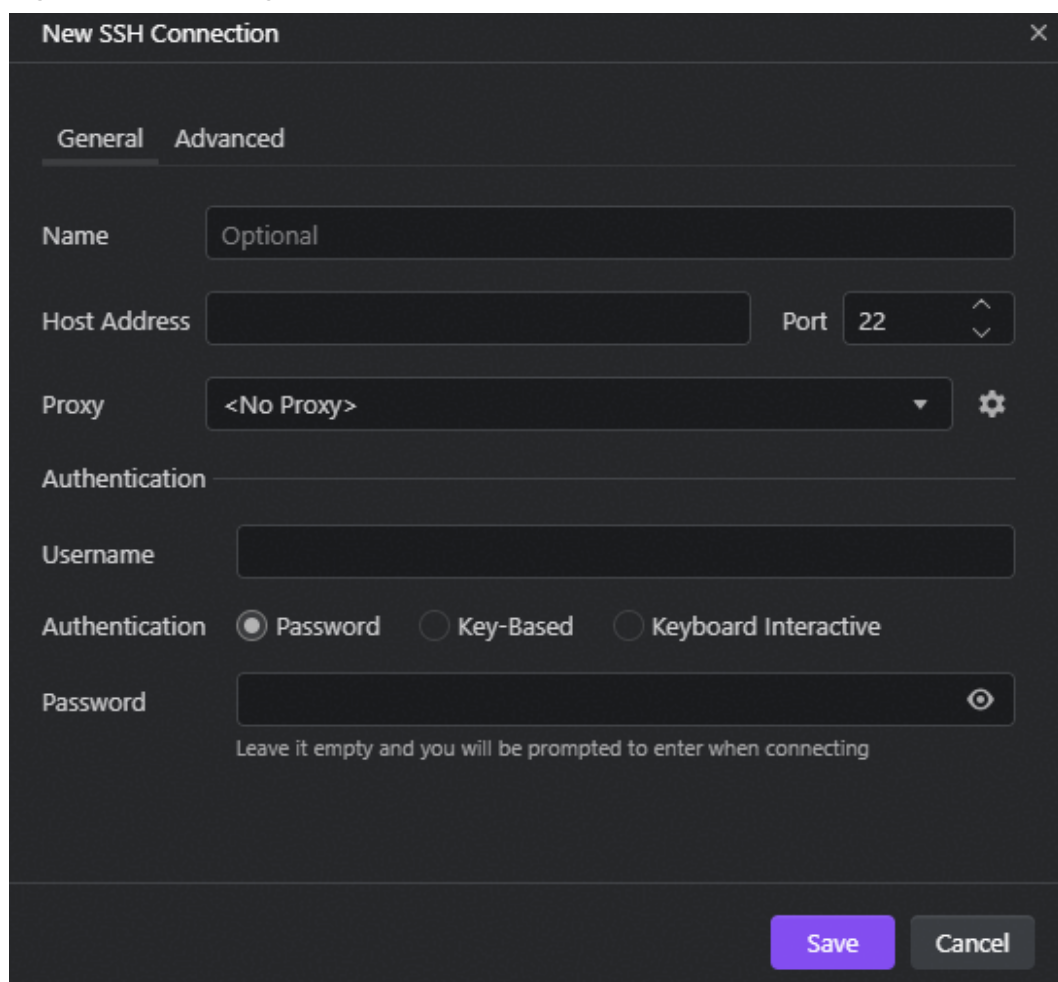
**Step 3** Under SSH hosts, click  to open the window for creating an SSH connection.

You can also click **Connect Current Window to Host** or **Connect New Window to Host** to quickly establish an SSH connection.

- Click **Connect Current Window to Host** to establish an SSH connection in the current IDE window.
- Click **Connect New Window to Host** to open a new IDE window and establish an SSH connection in it.

**Figure 12-2** Clicking the icon for adding an SSH host

**Step 4** Set the remote host information and click **Save**.

**Figure 12-3** Creating an SSH connection



**Table 12-2** Parameters for creating an SSH connection

Type	Parameter	Description
General	Name	Set an alias for the SSH connection for easy identification.
	Host Address	Enter the IP address of the target host to be connected.
	Port	Enter the TCP port number listened by the SSH service. The default value is 22.
	Proxy	Determine whether to connect to the target host by setting a proxy server.     • No proxy: Connect directly and skip all proxy settings.    • Application proxy: Select this option if you want to access the target host through a proxy server (for example, on a company intranet or restricted network). You can click  to access the detailed proxy settings page.    – Host address: IP address of the proxy server.    – Port: TCP port number listened by the proxy server.    – Proxy authentication: If the proxy server requires identity authentication, you need to enter the username and password.    – Check connection: Check whether the access to the target URL is normal with the current proxy configuration.
	Username	Enter the username for logging in to the target host.
	Authorization	Three authentication modes are provided: <b>Password</b> , <b>Key-based</b> , and <b>Keyboard Interactive</b> (two-factor authentication). You can choose any one of them.
	Password	Enter the password for logging in to the target host.  This parameter is displayed only when <b>Authentication</b> is set to <b>Password</b> or <b>Keyboard Interactive</b> .
	Private key	Click  and select the local SSH private key file (in the .pem, .ppk, or .key format).  This parameter is displayed only when <b>Authentication</b> is set to <b>Key-Based</b> .

Type	Parameter	Description
	Passphrase	If the private key file is encrypted, enter the decryption password. This parameter is displayed only when <b>Authentication</b> is set to <b>Key-Based</b> .
Advanced	Keep Alive	If this option is selected, the client periodically sends heartbeat packets to the server to check whether the connection is still valid. <b>Interval:</b> interval for sending keepalive messages, in seconds. Its default value is <b>30</b> .
	Jump Hosts	Determine whether to access the target host by configuring an intermediate server. This function is usually used for secure intranet access. After this function is enabled, the SSH connection is first established to the specified jump host, which then connects to the target host.

**Step 5** In the displayed dialog box, click **New Window** or **Current Window** to open the SSH connection in a new window or in the current window.

Wait until the connection is complete. If no error message is displayed and the connection information is displayed in the status bar in the lower left corner, the connection is successful.

**Step 6** After the connection is successful, click **Open Folder** in the explorer, browse and select the target folder on the remote device, and click **OK** to open it.

----End

## Managing Remote Connections

**Table 12-3** Managing remote connections

Operation	Instruction
Editing a remote connection	In the remote development of CodeArts Agent IDE, hover the cursor over the target remote connection and click  to modify configurations.
Disabling a remote connection	In the remote development of CodeArts Agent IDE, hover the cursor over the target remote connection and click  to disable the connection. After the connection is disabled, you can click  to reconnect.
Deleting a remote connection	In the remote development of CodeArts Agent IDE, hover the cursor over the target remote connection and click  to delete the connection from the list.

## Other Operations

**Table 12-4** Other operations

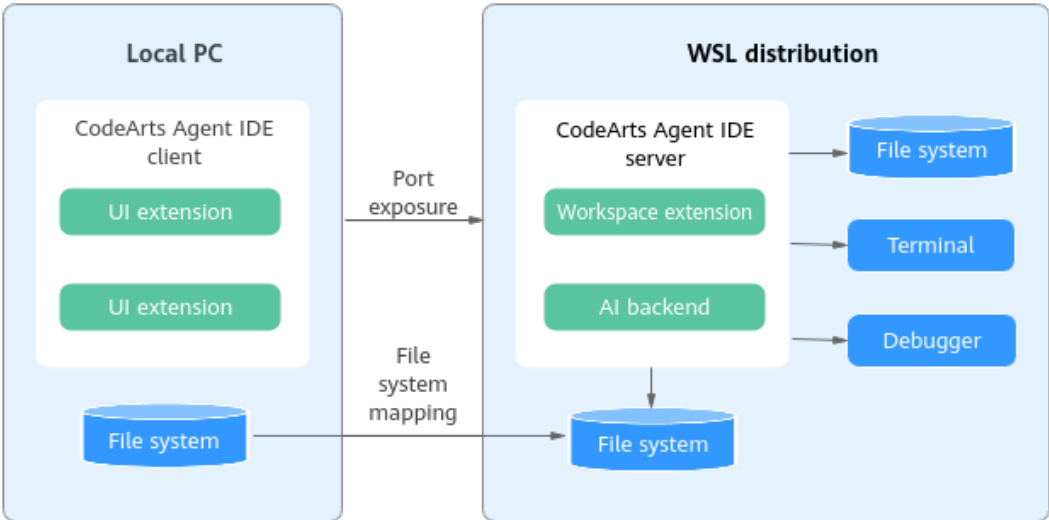
Operation	Description
	Expands all hierarchical levels under SSH hosts and unfolds all subnodes, allowing users to quickly browse and manage a large number of connection items.
	Collapses all expanded SSH host nodes to reduce the number of GUI layers and help users obtain a more neat view.
Displaying or hiding detected CodeArts projects/workspaces	Controls whether to display remote projects/workspaces and connection information automatically detected by the IDE and related to CodeArts.
Importing host settings	Supports importing SSH host configuration files (typically in .export format) from the local machine. This facilitates configuration migration between different computers or users, avoiding manual, repetitive entry of connection information to improve efficiency and consistency.
Exporting host settings	Exports current SSH host connection configurations to a .export file. This is used to back up your SSH configurations or quickly unify remote development environments within a team, achieving efficient and consistent remote connection setups.
Synchronizing information with local extensions	Triggers a synchronization of the current SSH host configuration to a local extension. This is typically used to ensure that extensions can properly recognize and use the latest SSH connection details, preventing connection issues caused by inconsistent configurations and improving stability and consistency.
Opening proxy manager	Opens the proxy server settings. If your network environment requires a proxy server to access remote hosts (such as an enterprise intranet), you can configure an HTTP proxy on this page to forward SSH connections through the proxy, ensuring successful connections to target hosts.
Opening jump host manager	Opens jump host configurations. When you cannot connect directly to the target server and require an intermediate server (a jump host), you can configure the relevant settings on this page to securely access the target server.
Showing connection logs	Views detailed SSH connection logs. When a connection fails or behaves abnormally, you can use these logs to troubleshoot issues and locate root causes.

Operation	Description
Setting log levels	Adjusts the verbosity level of SSH connection logs, such as Trace, Debug, and Info (default: Info).
Opening help documentation	Opens the official help documentation to access detailed guides on SSH host configuration and basic operations.

## 12.2 WSL Distributions

CodeArts Agent IDE supports remote connections to the Windows Subsystem for Linux (WSL). This function uses the local WSL environment as a remote development backend, enabling users on Windows to write, debug, and run code end-to-end within the local WSL environment, just like connecting to a remote server.

Figure 12-4 Remotely connecting to WSL



### Constraints

Table 12-5 Constraints

Category	Description
WSL	WSL has been installed on the local Windows system.

Category	Description
OS version	• glibc: 2.18 or later # Check the glibc version. ldd --version    • glibcxx: 3.4.22 or later # Check the glibcxx version. # Ubuntu strings /usr/lib/x86_64-linux-gnu/libstdc++.so.6   grep GLIBCXX # openEuler strings /lib64/libstdc++.so.6   grep GLIBCXX
Webpage preview	Direct opening of the browser for preview via the file:/// protocol is currently not supported.

## Connecting to WSL

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

**Step 2** Click  in the left menu bar of the IDE to open the remote development page.

**Step 3** Under WSL distributions, CodeArts Agent IDE automatically detects and identifies WSL environments existing in the current system.

**Step 4** Hover the cursor over the WSL environment to be connected, click , and wait until the connection is complete.

Wait until the connection is complete. If no error message is displayed and the connection information is displayed in the status bar in the lower left corner, the connection is successful.

**Step 5** After the connection is successful, click **Open Folder** in the explorer, browse and select the target folder on the WSL environment, and click **OK** to open it.

----End

## Managing WSL Connections

**Table 12-6** Managing WSL connections

Operation	Instruction
Disabling a remote connection	In the remote development of CodeArts Agent IDE, hover the cursor over the connected WSL environment and click  to disable it. After the connection is disabled, you can click  to reconnect.
Opening a folder in WSL	In the remote development of CodeArts Agent IDE, hover the cursor over the connected WSL environment, click  , and select the corresponding folder to open it.

Operation	Instruction
Displaying the remote host process	In the remote development of CodeArts Agent IDE, hover the cursor over the connected WSL environment and click  to view the process information of the remote host.

## Other Operations

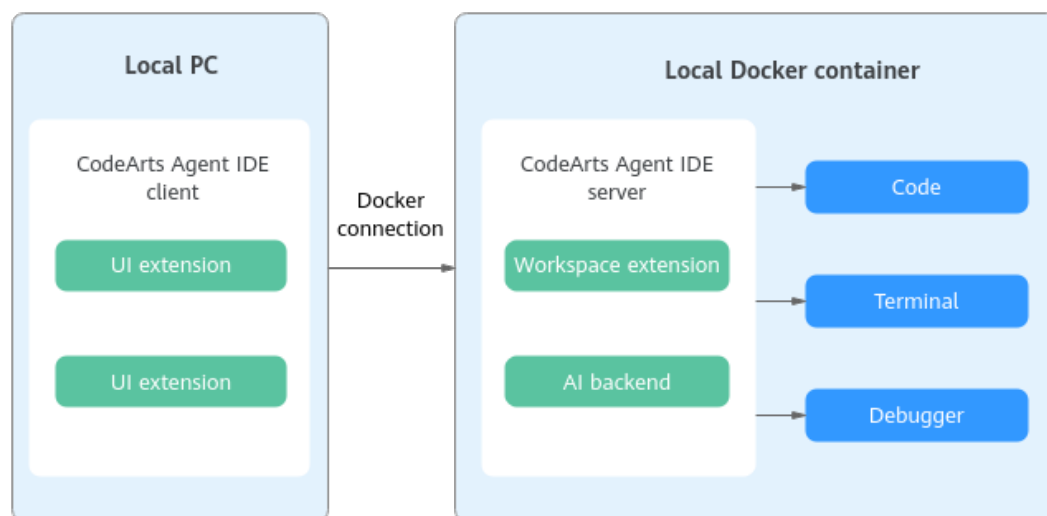
**Table 12-7** Other operations

Operation	Description
	Reloads the current WSL distribution list or status.
	Expands all hierarchical levels under WSL distributions and unfolds all subnodes, allowing users to quickly browse and manage a large number of connection items.
	Collapses all expanded WSL distributions, reducing the number of displayed levels and improving the readability and operation efficiency of the interface.
Displaying or hiding detected CodeArts projects	Controls whether to display remote projects and connection information automatically detected by the IDE and related to CodeArts.
Synchronizing information with local extensions	Triggers a synchronization of the current WSL environment configuration to a local extension. This is typically used to ensure that extensions can properly recognize and use the latest WSL distribution information, preventing connection issues caused by inconsistent configurations and improving stability and consistency.
Showing connection logs	Views detailed logs of WSL distribution connections. When a connection fails or behaves abnormally, you can use these logs to troubleshoot issues and locate root causes.
Setting log levels	Adjusts the verbosity level of WSL distribution connection logs, such as Trace, Debug, and Info (default: Info).
Opening help documentation	Opens the official help documentation to access detailed guides on WSL distribution configuration and basic operations.

## 12.3 Local Docker Container

CodeArts Agent IDE supports local Docker container connections. This function uses locally running Docker containers as development environment backends, enabling users to efficiently write, debug, and run code in containers via remote connections within a unified IDE interface. With this function, users can manage and develop code inside containers end-to-end without frequently switching terminals or manually entering the container.

**Figure 12-5** Remotely connecting to a local docker container



### Constraints

**Table 12-8** Constraints

Category	Description
Docker	Docker Desktop has been successfully deployed in the local Windows environment and is running properly.
OS version	- glibc: 2.18 or later # Check the glibc version. ldd --version   - glibcxx: 3.4.22 or later # Check the glibcxx version. # Ubuntu strings /usr/lib/x86_64-linux-gnu/libstdc++.so.6   grep GLIBCXX # openEuler strings /lib64/libstdc++.so.6   grep GLIBCXX
Webpage preview	Direct opening of the browser for preview via the file:/// protocol is currently not supported.

## Connecting to a Local Docker Container

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

**Step 2** Click  in the left menu bar of the IDE to open the remote development page.

**Step 3** Under Docker containers, CodeArts Agent IDE automatically detects and identifies existing Docker containers in the current system.

**Step 4** Hover the cursor over the Docker container to be connected, click , and wait until the connection is complete.

Before performing the connection operation, ensure that the Docker container is enabled. If the Docker container is not enabled, hover the cursor over the container and click  to enable it.

**Step 5** After the connection is successful, click **Open Folder** in the explorer, browse and select the target folder on the Docker container, and click **OK** to open it in the current window.

----End

## Managing Docker Container Connections

**Table 12-9** Managing Docker container connections

Operation	Instruction
Disabling a remote connection	In the remote development of CodeArts Agent IDE, hover the cursor over the connected Docker container and click  to disable it. After the connection is disabled, you can click  to reconnect.
Renaming	In the remote development of CodeArts Agent IDE, hover the cursor over the connected Docker container and click  to change the name of the remote connection.
Displaying the remote host process	In the remote development of CodeArts Agent IDE, hover the cursor over the connected Docker container and click  to view the process information of the remote host.

## Other Operations

**Table 12-10** Other operations

Operation	Description
	Reloads the current Docker container list or status.



Operation	Description
	Expands all hierarchical levels under Docker containers and unfolds all subnodes, allowing users to quickly browse and manage a large number of connection items.
	Collapses all expanded Docker containers, reducing the number of displayed levels and improving the readability and operation efficiency of the interface.
Displaying or hiding detected CodeArts projects	Controls whether to display remote projects and connection information automatically detected by the IDE and related to CodeArts.
Synchronizing information with local extensions	Triggers a synchronization of the current Docker container configuration to a local extension. This is typically used to ensure that extensions can properly recognize and use the latest Docker container details, preventing connection issues caused by inconsistent configurations and improving stability and consistency.
Showing connection logs	Views detailed Docker container logs. When a connection fails or behaves abnormally, you can use these logs to troubleshoot issues and locate root causes.
Setting log levels	Adjusts the verbosity level of Docker container logs, such as Trace, Debug, and Info (default: Info).
Opening help documentation	Opens the official help documentation to access detailed guides on Docker container configuration and basic operations.

# 13 Chat History

Chat history is CodeArts Agent's memory of past interactions, recording the chats between the user and the AI, including previous questions, answers, and context.

## Constraints

Table 13-1 Constraints

Category	Description
Function	Chat history is displayed by project. Do not modify project names arbitrarily, as doing so may render historical records unviewable.
Storage mode	Local storage. In the <b>opencode.db</b> file in the <b>%USERPROFILE%/.codeartsdoer/codearts-data</b> directory.

## Managing Historical Chats

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

**Step 2** On the chat panel of the CodeArts Agent, click  to view recent chat records.

### NOTE

If the chat panel of CodeArts Agent is not displayed, click  (**Expand AI Sidebar**) in the upper right corner of the top menu bar to open it.

**Step 3** You can view the latest **10** chat records and perform the following operations.

**Table 13-2** Managing historical chats

Operation	Instruction
Viewing all historical chats	<b>All</b> is displayed only when the number of historical chats is <b>no less than 10</b>.   In the lower left corner of the recent chat area, click <b>All</b> to view the historical chats of the current project and global chats (these chats do not belong to any project and are usually generated in earlier versions).
Viewing a historical chat	Click a chat record in the list to open it again on the chat panel.   <b>NOTE</b> If you need to review chats, search for content, share information, or organize data, you can view the chat history to access the complete information.
Editing a historical chat	Hover over  next to a chat and choose <b>Edit</b> to change the chat name.

Operation	Instruction
Sharing a historical chat	Sharing chat history is primarily applicable to scenarios such as saving and sharing chat content. For example, if your chat contains important information, you can export it for local backup to prevent data loss.   <b>Only chats in the completed state can be shared.</b>     - Method 1: Export a chat as a Markdown file.   - Hover the cursor over  next to the chat to be exported and choose <b>Share</b>.   - Select the content to be exported. You can select the check boxes to export partial chat content or choose <b>Select All</b> to export all chat content.   - Click <b>Export Markdown File</b> and select a local directory for saving the chat content. CodeArts Agent will export the chat content in Markdown format to the local PC. The file name is in the format of <b>CodeArts Agent YYYY-MM-DD.md</b>.       - Method 2: Export a chat as an image. <b>If the number of characters to be exported exceeds 10,000, they cannot be exported as an image.</b>   - Hover the cursor over  next to the chat to be exported and choose <b>Share</b>.   - Select the content to be exported. You can select the check boxes to export partial chat content or choose <b>Select All</b> to export all chat content.   - Click <b>Generate Image</b>. The image preview page is displayed.   - Click <b>Download</b> and select a local directory for saving the chat content. CodeArts Agent will export the chat content as an image to the local PC. The image name is in the format of <b>CodeArts Agent YYYY-MM-DD.png</b>.  You can also copy the image to tools such as Word and PowerPoint.

Operation	Instruction
Deleting chats one by one	When there are few chat records, deleting them individually is not only more intuitive and flexible but also allows for the precise removal of specific content.      1. Hover the cursor over  next to the chat you want to delete and select <b>Delete</b>.    2. In the displayed dialog box, click <b>OK</b>.      <b>CAUTION</b> Deleted chats cannot be recovered. Exercise caution when performing this operation.
Batch deleting chats	Deleting many chats one by one manually is inefficient; you can also batch delete chats using the following steps: For example, if a large number of temporary chats were generated while debugging or testing the agent, you can use the batch delete feature to clear these unneeded records with one click.   In the <b>agent</b> mode, if there is a chat in progress, batch deletion will not allow the removal of chats that are in <b>intelligent Q&amp;A</b> mode.      1. In the lower left corner of the recent chat page, click <b>All</b>.    2. Click  next to the search box.    3. In the chat list, select the records to be deleted and click <b>Delete</b>. To delete all records, scroll the list to the earliest record, choose <b>Selected</b> on the top, and click <b>Delete</b>.    4. In the displayed dialog box, click <b>OK</b>.      <b>CAUTION</b> Deleted chats cannot be recovered. Exercise caution when performing this operation.

----End

## Related Documents

Modifying a project name will render the chat history of that project unviewable in CodeArts Agent IDE. If you need to restore your chat history, follow the instructions in What Should I Do If I Cannot View Chat History in the Original Project?

# 14 Hooks

With hooks, you can insert custom logic into key lifecycle nodes of CodeArts Agent IDE and extensions to extend functionality without modifying any existing code.

Unlike prompts that rely on model understanding, hook operations yield deterministic outcomes. Once an event is triggered, the designated script executes reliably and consistently, without being affected by model understanding discrepancies.

## Constraints

Table 14-1 Constraints

Category	Description
Function	Hook scripts currently do not support real-time loading; a reload is required before they can be used.
Language	Hook scripts support only JavaScript and TypeScript.
External dependencies	If external dependency packages are used, you must create a <b>package.json</b> file in the configuration directory where the hook is located and declare the required dependencies.

## Supported Hook Events

Table 14-2 Supported hook events

Event Category	Event Name	Trigger Time	Blockable	Typical Application Scenario	Description
Chat message	chat.message	Triggered after a user sends a message but before it enters the processing workflow.	Yes	Chat message processing	Automatically triggered when a new message arrives. Supports editing message content and parts.
Chat parameter	chat.params	Triggered before each LLM call.	Yes	Chat parameter modification	Supports dynamically adjusting LLM inference parameters (including temperature, topP, topK, etc.) to optimize model output as needed.
Chat request header	chat.headers	Triggered before each LLM call.	Yes	Chat request header modification	Supports customizing and modifying HTTP request header fields to adapt to complex requirements such as network proxies and interface authentication.

Event Category	Event Name	Trigger Time	Blockable	Typical Application Scenario	Description
Chat response	chat.response	Triggered after the LLM returns a response.	No	Chat response logging	Automatically captures core metrics of the model response, such as tokens, cost, and duration, achieving precise billing statistics, efficient troubleshooting, and complete log retention.
Chat error	chat.error	Triggered when an error occurs during an LLM call.	No	Chat error logging	Logs error details when an LLM call fails.
Chat compression	chat.compression	Triggered during context compression.	No	Chat compression logging	Logs the token counts before and after context compression to assist in storage and performance tuning.
Chat end	chat.finished	Triggered when a chat ends.	No	Chat end logging	Logs the chat end event, including the token consumption and running information of the current run.



Event Category	Event Name	Trigger Time	Blockable	Typical Application Scenario	Description
User approval	user.approval	Triggered when a user confirmation or approval action is required.	Yes	User approval logging	Intercepts high-risk operations and initiates manual confirmation and approval, controlling the execution permissions of sensitive operations.
Turn end	turn.end	Triggered when each chat turn ends.	No	Turn end logging	Logs the end of each chat turn, containing key information such as processing duration and termination reasons.
Command execution	command.execute.before	Triggered before a command is executed.	Yes	Command preprocessing	Supports custom configuration of parts to achieve command preprocessing and flexible adaptation.
Tool execution	tool.execute.before	Triggered before a tool is executed.	Yes	Tool execution pre-parameter adjustment	Supports customizing and modifying args before tool execution to achieve validation, dynamic rewriting, and service adaptation.

Event Category	Event Name	Trigger Time	Blockable	Typical Application Scenario	Description
	tool.execute.after	Triggered after a tool is executed.	Yes	Tool execution post-result processing	Supports customizing and modifying returned results, including the title, output, and metadata fields.
Tool definition	tool.definition	Triggered before a tool definition is sent to the LLM.	Yes	Tool definition modification	Supports customizing configuration descriptions and parameters before tool definitions are pushed to the model, ensuring precise adaptation to service scenarios.
Shell environment	shell.env	Triggered when fetching Shell environment variables.	Yes	Shell environment variable configuration and management	Supports adding or modifying environment variables when Shell environment variables are fetched.
Permission request	permission.ask	Triggered when permission verification is required.	Yes	Permission request processing	Processes permission validation requests and supports configuring three strategies: allow, deny, and ask.

Event Category	Event Name	Trigger Time	Blockable	Typical Application Scenario	Description
Message conversion	experimental.chat.messages.transform	Triggered before the message list is sent to the LLM.	Yes	Message conversion	Supports preprocessing the chat list before it is reported to the model.
System message conversion	experimental.chat.system.transform	Triggered before the system prompt is sent.	Yes	System message conversion	Supports optimizing prompt content before the prompt is sent.
Session compression	experimental.session.compacting	Triggered before context compression starts.	Yes	Chat compression	Supports customizing the compression prompt before context compression is executed.
Text completion	experimental.text.complete	Triggered when text completion finishes.	Yes	Text completion result customization	Supports modifying the output content after text completion is completed.
Configuration loading	config	Triggered when application startup configurations are loaded.	No	Configuration loading	Performs configuration initialization and dynamic parameter injection and is triggered during the loading of custom configurations.
Common event	event	Triggered when all Bus events are subscribed to.	No	Event processing	Processes common events.

## Storage Paths for Plug-ins and Dependency Files

A plug-in is a JavaScript/TypeScript module that operates by exporting plug-in functions. Each function receives a context object and returns the corresponding hook object. If external packages are required within a plug-in file, you must create a **package.json** file in the configuration directory and configure the necessary dependencies. CodeArts Agent automatically installs these dependencies when it starts.

**Table 14-3** Plug-in file loading paths

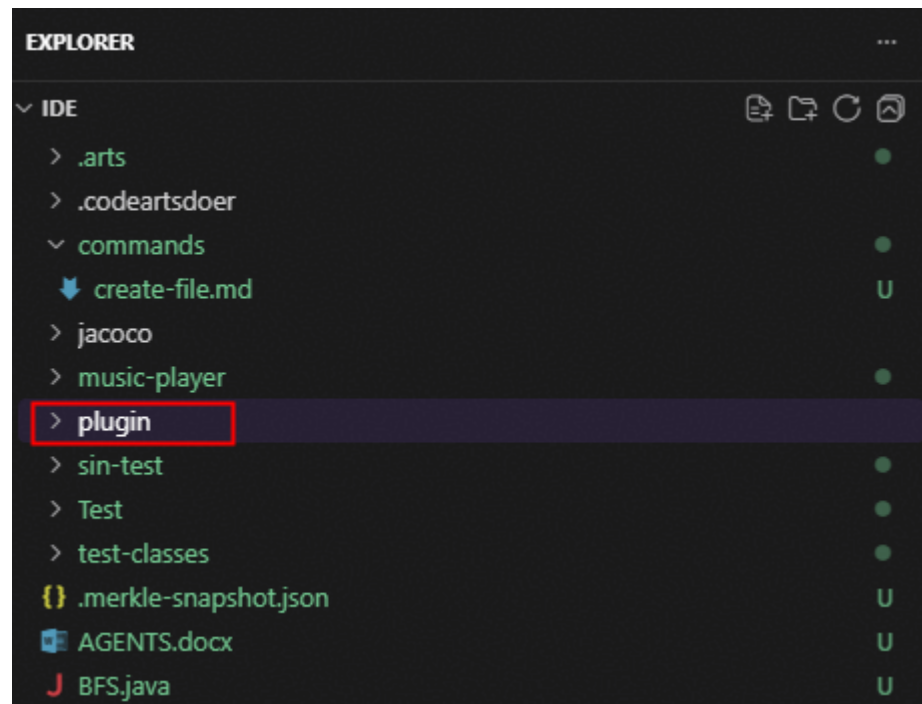
Plug-in Type	Plug-in File Storage Path	Plug-in Dependency Storage Path	Description
Project	Root directory of the current project: <b>./codeartsdoer/plugin</b> or <b>./codeartsdoer/plugins</b>	Root directory of the current project: <b>./codeartsdoer/</b>	Valid only for the current project.
User	Local <b>%USERPROFILE%/.codeartsdoer/plugin</b> or <b>%USERPROFILE%/.codeartsdoer/plugins</b>	Local <b>%USERPROFILE%/.codeartsdoer/</b>	Valid for all projects of the current user.

## Quick Start

The following example demonstrates how to filter sensitive information, such as passwords and keys, from user inputs to effectively prevent data leaks.

**Step 1** Create a directory for storing the plug-in file.

Create a directory named **plugin** in the project root directory **./codeartsdoer**.

**Figure 14-1** Creating the plugin directory**Step 2** Create a script.

In the **plugin** directory, create a new file named **sensitiveInfoFilteringPlugin.ts**, write the hook logic code in the file, and save the file.

```
// Import the plug-in type to define the plug-in.
import type { Plugin } from "@opencode-ai/plugin"

// Export the plugin implementation. Plugin is an asynchronous function that receives plugin input
// parameters and returns a hook object.
export const SensitiveInfoFilteringPlugin: Plugin = async ({}) => {
 // Return a hooks object containing various hook functions.
 return {
 // Listen to the hook after the tool is executed. This function is triggered after the tool execution is
 // complete.
 "tool.execute.after": async (
 // Input parameters include the tool name, session ID, call ID, inference ID, and other information.
 input: { tool: string; sessionId: string; callId: string; args: any },
 // Output parameters include the title, output, and metadata of the tool execution result.
 output: {
 title: string
 output: string
 metadata: any
 },
) => {
 // Check if the operation is to read a file (tool name is "read") and the output content exists.
 if (input.tool === "read" && output.output) {
 // Sensitive information matching patterns (supporting JSON and other common formats).
 // Matches "password": "value" or password: value.
 // First regex: matches JSON format, such as "password": "****".
 // Second regex: matches common key-value formats, such as password=*** or password:***.
 const sensitivePatterns = [
 // Matches sensitive fields in JSON format: double-quoted key + colon + double-quoted value.
 new RegExp(`"password|passwd|pwd|api[_-]?key|secret|token|key|password"\\s*:\\s*"([^\"]+)"`, "g"),
 // Matches sensitive fields in common format: key + colon or equal sign + unquoted value.
 new RegExp(`(password|passwd|pwd|api[_-]?key|secret|token)\\s*[:=]\\s*"([^\s\\\\\\\\]+)"`, "gi"),
]
 // Assign the tool output content to the local variable content for subsequent processing.
```

```
let content = output.output

// Traverse all sensitive information matching patterns and replace them one by one.
for (const pattern of sensitivePatterns) {
 // Use the replace method for substitution, passing a callback function to handle match results.
 // match: the fully matched string.
 // key: capture group 1, i.e., the sensitive field name (password, api_key, etc.)
 // value: capture group 2, i.e., the actual content of the sensitive field (e.g., actual password or key)
 content = content.replace(pattern, (match, key, value) => {
 // If sensitive information is matched, replace the actual value with the [Sensitive Info] placeholder.
 return match.replace(value, "[Sensitive Info]")
 })
}
// Update the output field in the output object to write back the filtered content.
output.output = content
}
```

**Step 3** Restart the IDE.

In the upper left corner of CodeArts Agent IDE, click **File** and choose **Restart IDE**. Because the current hook script does not support real-time loading, you need to restart the IDE for the script to take effect.

**CAUTION**

To ensure that the script takes effect, it is advised to terminate all Bun processes in the task manager to completely clear any remaining instances.

**Step 4** Verify the effect.

After you enter a prompt in the input box, the sensitive information in the results returned by the model will be automatically masked.

----End

## Creating Hooks

**Step 1** Select an event.

Select an event name in [Supported Hook Events](#) based on service requirements.

**Step 2** Compile the hook script.

1. Create a plug-in file.

Create a .ts or .js file in the directory described in [Storage Paths for Plug-ins and Dependency Files](#).

2. Compile the hook logic.

In [Hook Event Reference](#), locate **// Hook implementations go here**, and write your specific hook implementation code there based on your actual service requirements.

**Step 3** (Optional) Configure dependencies.

If external packages are required within a plug-in file, you must create a **package.json** file in the configuration directory described in [Storage Paths for Plug-ins and Dependency Files](#) and configure the necessary dependencies.

```
{
 "dependencies": {
 "Dependency package name 1": "Version number",
 "Dependency package name 2": "Version number"
 }
}
```

The following is an example of the **package.json** file:

```
{
 "dependencies": {
 "@opencode-ai/plugin": "*",
 "@babel/core": "7.29.0",
 "@langfuse/otel": "4.5.1"
 }
}
```

**Step 4** Restart the IDE to verify the effect.

----End

## Hook Event Reference

Examples of supported hook events are provided below. Write your specific implementation logic at the **// Hook implementations go here** comment. Once the code is written, it can be used directly.

### chat.message

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "chat.message": async (
 input: { sessionId: string; agent?: string; model?: { providerID: string; modelID: string }; messageID?: string; variant?: string },
 output: { message: any; parts: any[] }
) => {
 // Hook implementations go here
 }
 }
}
```

### chat.params

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "chat.params": async (
 input: { sessionId: string; agent: string; model: any; provider: any; message: any; inferenceID?: string; isEnsureTitle?: boolean },
 output: { temperature: number; topP: number; topK: number; options: Record<string, any> }
) => {
 // Hook implementations go here
 }
 }
}
```

### chat.headers

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "chat.headers": async (
 input: { sessionId: string; agent: string; model: any; provider: any; message: any },
 output: { headers: Record<string, string> }
) => {

```

```
 // Hook implementations go here
 }
}
```

## chat.response

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "chat.response": async (
 input: {
 sessionID: string
 parentID: string
 messageID: string
 userMessageID: string
 model: { id: string; providerID: string }
 agent: string
 },
 output: {
 finishReason: string
 tokens?: { input: number; output: number; reasoning: number; cache?: {write:number; read:number} }
 cost?: number
 duration: number
 responseText: string
 firstTokenTime: number
 },
) => {
 // Hook implementations go here
 }
 }
}
```

## chat.error

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "chat.error": async (
 input: { sessionID: string; messageID: string; userMessageID: string; model: { id: string; providerID:
string }; agent: string },
 output: { error: Error; duration: number },
) => {
 // Hook implementations go here
 }
 }
}
```

## chat.compression

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "chat.compression": async (
 input: { sessionID: string, tokensBefore: string, tokensAfter: string },
 output: { },
) => {
 // Hook implementations go here
 }
 }
}
```

## chat.finished

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "chat.finished": async (
 input: { sessionID: string, agentRunID: string, agentID: string, loopStartTime: string, tokensInput: string,
tokensOutput: string },
 output: { },
) => {
 // Hook implementations go here
 }
 }
}
```



```
}
}
}
```

## user.approval

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "user.approval": async (
 input: {requestID: string, reply: string, sessionID: string},
 output: {}
) => {
 // Hook implementations go here
 }
 }
}
```

## turn.end

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 /** The trigger hook is used to record turn end logs. */
 "turn.end": async (
 input: {sessionID: string, agentID: string, turnCount: string, durationMs: string, terminateReason: string},
 output: {}
) => {
 // Hook implementations go here
 }
 }
}
```

## command.execute.before

```
import type { Part } from "@opencode-ai/sdk"

export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "command.execute.before": async (
 input: { command: string; sessionID: string; arguments: string },
 output: { parts: Part[] },
) => {
 // Hook implementations go here
 }
 }
}
```

## tool.execute.before

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "tool.execute.before": async (
 input: { tool: string; sessionID: string; callID: string; inferenceID?: string },
 output: { args: any },
) => {
 // Hook implementations go here
 }
 }
}
```

## tool.execute.after

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "tool.execute.after": async (
 input: { tool: string; sessionID: string; callID: string; args: any },
 output: {
 title: string
 output: string
 }
)
 }
}
```

```
 metadata: any
 },
) => {
 // Hook implementations go here
}
}
```

## tool.definition

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "tool.definition": async (
 input: { toolID: string },
 output: { description: string; parameters: any }
) => {
 // Hook implementations go here
 }
 }
}
```

## shell.env

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "shell.env": async (
 input: { cwd: string; sessionId?: string; callID?: string },
 output: { env: Record<string, string> },
) => {
 // Hook implementations go here
 }
 }
}
```

## permission.ask

```
import type { Permission } from "@opencode-ai/sdk"

export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "permission.ask": async (
 input: Permission,
 output: { status: "ask" | "deny" | "allow" }
) => {
 // Hook implementations go here
 }
 }
}
```

## experimental.chat.messages.transform

```
import type { Message, Part } from "@opencode-ai/sdk"

export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "experimental.chat.messages.transform": async (
 input: {},
 output: {
 messages: {
 info: Message
 parts: Part[]
 }[]
 },
) => {
 // Hook implementations go here
 }
 }
}
```

### experimental.chat.system.transform

```
import type { Model } from "@opencode-ai/sdk"

export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "experimental.chat.system.transform": async (
 input: { sessionId?: string; model: Model },
 output: {
 system: string[]
 }
) => {
 // Hook implementations go here
 }
 }
}
```

### experimental.session.compacting

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "experimental.session.compacting": async (
 input: { sessionId: string },
 output: { context: string[]; prompt?: string },
) => {
 // Hook implementations go here
 }
 }
}
```

### experimental.text.complete

```
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 "experimental.text.complete": async (
 input: { sessionId: string; messageId: string; partID: string },
 output: { text: string },
) => {
 // Hook implementations go here
 }
 }
}
```

### config

```
import type { Config } from "@opencode-ai/sdk"
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 config: async (
 input: Config
) => {
 // Hook implementations go here
 }
 }
}
```

### event

```
import type { Event } from "@opencode-ai/sdk"
export const MyPlugin = async ({ project, client, $, directory, worktree }) => {
 return {
 event: async (
 input: { event: Event }
) => {
 // Hook implementations go here
 }
 }
}
```

## Usage Example

Taking "displaying a pop-up window to prompt for high-risk operations before deleting a file" as an example, this section provides a detailed guide on how to use a hook. This example uses a project-level configuration within the plug-in directory.

### Step 1 Select an event based on the description in [Supported Hook Events](#).

Deletions of files will call a tool, usually the deleteFile tool. Because the pop-up window needs to appear before the file is deleted, you should select the tool.execute.before event here.

### Step 2 Compile the hook script.

1. Create a directory named **plugin** in the project root directory **./ .codeartsdoer**.
2. In the **plugin** directory, create a file named **deleteFileHintPlugin.ts**.
3. Compile the script and save the file.

```
// Import the type definition of the OpenCode plug-in.
import type {Plugin} from "@opencode-ai/plugin";
// Import the Node.js child process module to execute system commands.
import { exec } from "child_process";
// Import the promisify function from the util module to convert callback functions into Promises.
import { promisify } from "util";
// Convert the exec callback function into a Promise-based asynchronous function.
const execAsync = promisify(exec);
// Define and export the plug-in, accepting a client parameter to interact with OpenCode.
export const DeleteFileHintPlugin: Plugin = async ({ client }) => {
 // Return the hook collection of the plug-in.
 return {
 // Listen for the hook for tool.execute.before.
 "tool.execute.before": async (
 // Input parameters: tool name, session ID, call ID, and inference ID.
 input: { tool: string; sessionId: string; callId: string; inferenceId?: string },
 // Output parameters: parameters for tool calling.
 output: { args: any }
) => {
 // Check if the operation is to delete a file.
 if (input.tool === "deleteFile") {
 // Define the message content for the warning pop-up window.
 const message = `Deleting files is a high-risk operation. Exercise caution when performing this operation.`
 // Execute commands based on the operating system.
 if (process.platform === "win32") {
 // Windows: Use PowerShell to call the system MessageBox.
 const cmd = `powershell -NoProfile -Command "Add-Type -AssemblyName System.Windows.Forms; [System.Windows.Forms.MessageBox]::Show('${message}', 'Warning', 'OK', 'Warning')"`
 await execAsync(cmd)
 } else if (process.platform === "darwin") {
 // macOS: Use osascript to call the system pop-up window.
 const cmd = `osascript -e 'display alert "Warning" message "${message}"'`
 await execAsync(cmd)
 } else if (process.platform === "linux") {
 // Linux: Use notify-send to send a desktop notification.
 const cmd = `notify-send "Warning" "${message}"`
 await execAsync(cmd)
 }
 }
 }
 }
}
```

### Step 3 Restart the IDE to verify the effect.

1. In the upper left corner of CodeArts Agent IDE, click **File** and choose **Restart IDE**. Because the current hook script does not support real-time loading, you need to restart the IDE for the script to take effect.



To ensure that the script takes effect, it is advised to terminate all Bun processes in the task manager to completely clear any remaining instances.

2. Enter **Delete file \*\*\*** in the text box. A warning window is displayed on the client.

\*\*\* indicates the name of the file to be deleted.

----End

# 15 Slash Commands

---

Slash commands are an efficient way to interact, allowing the agent to quickly execute common tasks within a chat. CodeArts Agent has been preconfigured with some common commands. You can also customize commands as needed.

Through custom commands, you can encapsulate frequently used operations or complex instructions into simple commands, thereby simplifying your workflow and enhancing collaboration efficiency with the AI. In CodeArts Agent, using custom slash commands offers the following advantages:

- **Reusing common prompts:** Boosting efficiency and reducing repetitive typing  
Encapsulate frequently used prompts into shortcut commands to avoid typing the same instructions manually every time. For example, a custom command `/code_explain` can be used to explain the functionality, design mindset, and dependencies of the selected code.
- **Standardizing output formats:** Unifying team standards and decreasing collaborative friction  
Enforce the AI to generate outputs that align with team standards through preset templates or rules, ensuring consistency. For example, a custom `/api_doc_format` can be used to convert API descriptions into a unified Markdown or Swagger format, making it easier for frontend integration or document platform synchronization.
- **Automating common development workflows:** Triggering multi-step tasks with one click  
Package multiple operations or analysis steps into a single command to achieve "one-click trigger, end-to-end execution." For example, a custom command `/code_review` can be used to automatically scan the code of the current branch, identify potential bugs, performance bottlenecks, or style inconsistencies, and generate a comprehensive report.

## Built-in Slash Commands

CodeArts Agent has been preconfigured with the commands listed in [Table 15-1](#). You can directly invoke them without additional configuration.

**Table 15-1** Built-in slash commands

Mode	Command	Description
Intelligent Q&A	/optimize	Code optimization Allows CodeArts Agent to analyze code performance bottlenecks and provide optimizations, helping users improve the readability and maintainability of code.
	/comment	Code commenting Allows the system to analyze the code's architecture and logic, then auto-generate clear, meaningful comments. It facilitates standardizing the comment style across your entire repository, so readability becomes second nature and code archaeology is a thing of the past.
	/explain	Code explanation Allows the system to walk you through complex code with function, purpose, application scenario, and main logic. This clear explanation helps you get up to speed faster, and make knowledge flows smoothly, keeping the whole team in sync.
	/tests	Unit test Powers up your coding with generated unit test cases that slot into various test frameworks, such as JUnit, JMockit, pytest, unittest, and GTest. From single method to comprehensive class-wide tests, the code quality is guaranteed and defect risks are reduced.
	/help	Help Displays shortcut operation tips.
	/clear	Clear screen Clears the Q&A window content and opens a new one.
Agent Custom agent	/clear	Clear screen Clears the agent window content and opens a new one.
	/init	Agent.md one-click initialization Automatically generates the specification configuration file <b>Agent.md</b> for stock projects.

Mode	Command	Description
	/review	Code review Initiates a review on code, automatically identifying and fixing issues.
	/sdd-new	Creation of requirement specifications Generates the requirement specification document <b>spec.md</b> based on requirement descriptions.
	/sdd-design	Creation of SDD technical design Generates <b>design.md</b> based on <b>spec.md</b> .
	/sdd-tasks	Creation of SDD coding tasks Generates the task list <b>tasks.md</b> based on <b>spec.md</b> and <b>design.md</b> .
	/sdd-apply	Execution of SDD coding tasks Executes coding tasks one by one based on <b>tasks.md</b> .
Agent	/bugfix	Automatically invokes the repair expert agent to fix code issues based on the descriptions in the trouble tickets submitted by users. For details, see <a href="#">/bugfix</a> .
Agent	/fix-review	Fixes a single piece of code based on the review comment from the code review agent. For details, see <a href="#">/fix-review</a> .
Agent	/fix-review-batch	Fixes multiple pieces of code based on the review comments from the code review agent. For details, see <a href="#">/fix-review-batch</a> .
Agent	/fix-codeartscheck-batch	Batch fixes alert issues generated by CodeArts Check. For details, see <a href="#">/fix-codeartscheck-batch</a> .

## Custom Slash Commands

CodeArts Agent allows you to create a Markdown file in the **commands/** directory under a specified path to customize slash commands. After the command is loaded successfully, you can run it in CodeArts Agent by using **/**. **Custom slash commands can be used only in agent mode.**



**Table 15-2** Categories of custom slash commands

Command Type	Command File Storage Path	Priority and Overriding Relationship	Description
Project	Root directory of the current project: <code>./codeartsdoer/commands/</code>	The priority of a command depends on its loading sequence. Commands loaded later have higher priority and will override previously loaded commands with the same name.	Valid only for the current project.
User	Local <code>%USERPROFILE%/.codeartsdoer/commands/</code>		Valid for all projects of the current user.

 **NOTE**

User-level configurations apply globally to all projects. Project-level configurations take effect only to the current project. Both can be used simultaneously, but care must be taken regarding potential naming conflicts.

## Custom Project Slash Commands

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

**Step 2** Create a dedicated directory named **commands** to store related command files.

In the explorer, right-click the **.codeartsdoer** directory of the corresponding project and choose **New Folder**. Enter **commands** in the displayed input box.

**Step 3** Create a Markdown file to define commands.

1. Right-click the created **commands** directory and choose **New File** from the shortcut menu to create a Markdown file.

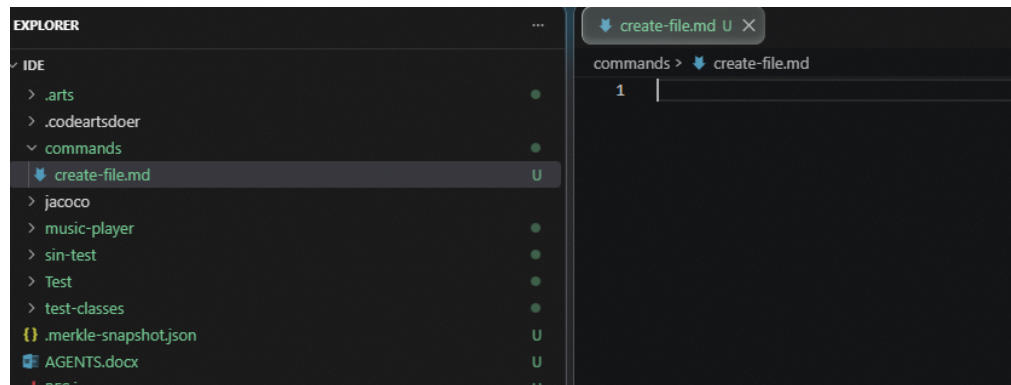
To ensure clear mapping between commands and functions, name Markdown files according to the following rules:

- File name as the slash command

The file name of a Markdown file should be identical to its corresponding slash command. For example, if the file creation function is executed via the **/create-file** command, the Markdown file corresponding to this function should be named **create-file.md**.

- Single file for a single function

Each Markdown file should correspond to only one function. Avoid mixing descriptions of multiple functions within a single file to keep the content concise.

**Figure 15-1** Creating create-file.md

2. Define the file content and save it.

In CodeArts Agent, you can use the **\$ARGUMENTS** placeholder to capture all parameters passed to the command, or use positional parameters like **\$N** to accurately access a specific parameter. For example, if you want to define a command that creates a file based on specific parameter content, the configuration example for **create-file.md** is as follows:

```

description: Create a new file with the specified content

Create a file named $1 in the directory $2,
with the following content:
$3
```

In the preceding example, three location parameters are defined. **\$1** indicates the name of the file to be created, **\$2** indicates the file storage path, and **\$3** indicates the file content.

**Step 4** After the file is saved, restart the IDE for the configuration to take effect.

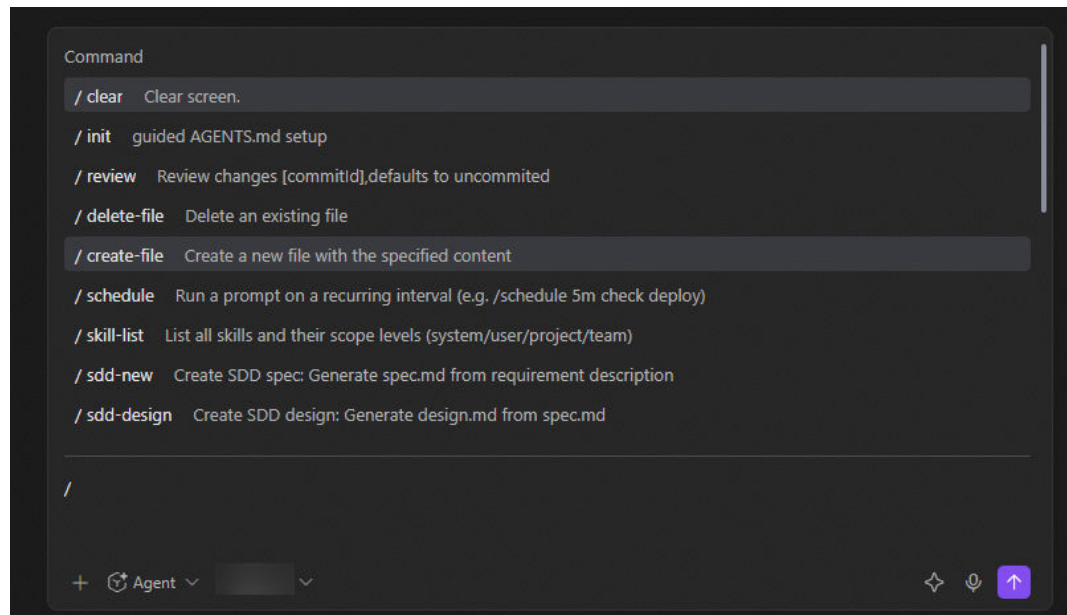
Restart CodeArts Agent IDE. After login, open the chat panel. Switch to the agent mode and enter a **/** in the input box. The custom project-level command **/create-file** is displayed in the shortcut menu.

---

**CAUTION**

If the target command is not displayed, terminate the Bun processes in the task manager, and then re-login to CodeArts Agent IDE.

---

**Figure 15-2** Displaying the custom project-level command `/create-file`

----End

## Custom User Slash Commands

**Step 1** Create a Markdown file to define commands.

Windows 11 (x64) is used as an example.

1. Create a **commands** folder in the local **%USERPROFILE%/.codeartsdoer/** directory (**C:\Users\Your username\.codeartsdoer**).
2. Go to the **commands** directory and create a Markdown file named **delete-file.md**.
3. Define the file content and save it.

In CodeArts Agent, you can use the **\$ARGUMENTS** placeholder to capture all parameters passed to the command, or use positional parameters like **\$N** to accurately access a specific parameter. For example, if you want to define a command that deletes a file based on specific parameter content, the configuration example for **delete-file.md** is as follows:

```

description: Delete an existing file

```

Delete the file named **\$1** located in the directory **\$2**.

**\$1** indicates the name of the file to be deleted, and **\$2** indicates the path of the file.

**Step 2** Restart CodeArts Agent IDE. After login, open the chat panel. Switch to the agent mode and enter a **/** in the input box. The custom user-level command **/delete-file** is displayed in the shortcut menu.

**CAUTION**

If the target command is not displayed, terminate the Bun processes in the task manager, and then re-login to CodeArts Agent IDE.

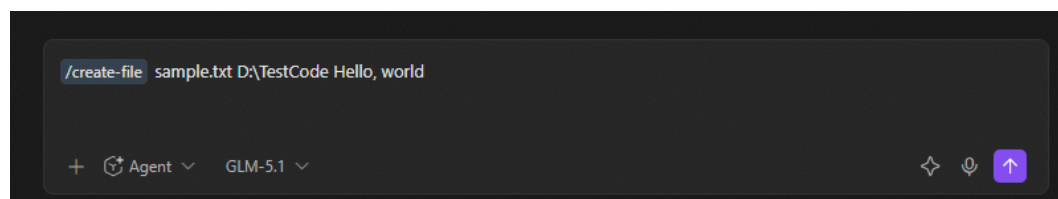
----End

## Using Custom Commands

- Step 1** Log in to CodeArts Agent IDE as instructed in the [quick start](#).
- Step 2** At the bottom of the input box on the chat panel, choose **Agent** to switch to the agent mode.
- Step 3** In the input box, enter a / and choose the custom command **/create-file** from the shortcut menu.
- Step 4** Enter the command and click .

In the preceding example, **create-file.md** defines three location parameters: file name (\$1), storage path (\$2), and file content (\$3). You need to pass specific values for these parameters when using the command.

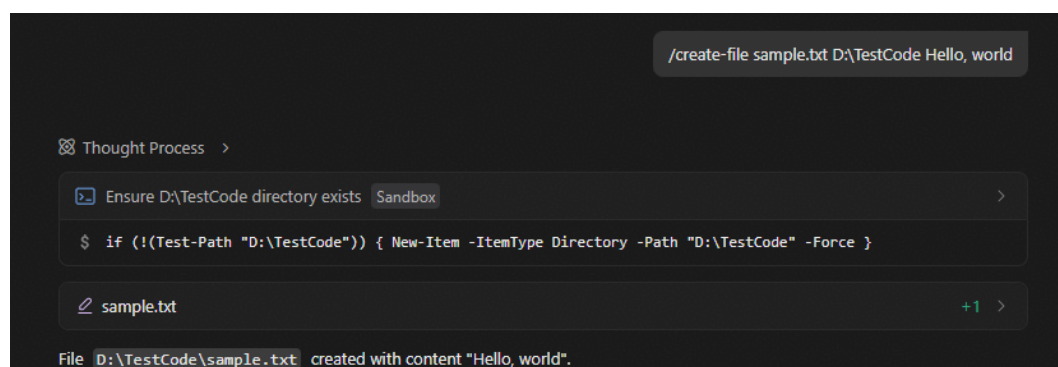
**Figure 15-3** Specifying parameter values

**NOTE**

You only need to enter parameters here if they are defined in the Markdown file; separate multiple parameters with spaces. If the number of parameters entered exceeds the number defined in the file, the excess part will be merged into the content of the last parameter.

- Step 5** After the command is executed, go to the specified directory (for example, **D:\TestCode**) and check whether the file and its content meet the expectation.

**Figure 15-4** File created



In the specified directory, you can view the generated file **sample.txt**, whose content is "Hello, world!"

----**End**

# 16 Intelligent Programming

## 16.1 Code Review

CodeArts Agent IDE automates code reviews by efficiently coordinating two core commands, `/codeagent-review-rules` and `/review`, offering precise and powerful code quality control.

### Constraints

Table 16-1 Constraints

Category	Description
Environment requirements	The <code>/review</code> command must be executed in a Git repository whose working directory contains staged, unstaged, or untracked file changes.
Supported file formats	Only code files in the supported programming languages can be reviewed. Non-code files (such as .json, .md, and .css) are automatically excluded from the review process.    - JavaScript/TypeScript: .ts, .tsx, .js, .jsx, .mjs, .cjs, .mts, .cts   - Python: .py, .pyi, .pyw, .pyx   - Java: .java   - C/C++: .c, .cpp, .cc, .cxx, .h, .hpp, .hh, .hxx   - Go: .go

### Command Usage

The following sections describe how to use the two core commands: `/review` and `/codeagent-review-rules`.

**/review**

**Function:** Starts the code review process to comprehensively review the current code changes or those in a specified commit.

**Syntax:** Select the syntax that suits your needs and press **Enter** to trigger a comprehensive review of the specified code changes.

**Table 16-2** /review command syntax

Syntax	Description
/review	Reviews all code changes in the current working directory, including staged changes, unstaged changes, and newly untracked files.
/review <commitId>	Reviews all code changes (separated by spaces) in a single commit.
/review commitId:<commitId>	Reviews all code changes (separated by colons) in a single commit.
/review commitId=<commitId>	Reviews all code changes (separated by equal signs) in a single commit.
/review <i>Review the code changes in &lt;commitId&gt;</i>	Reviews a specified commit using a natural language description.

**Table 16-3** /review command parameters

Parameter	Type	Description
commitId	String (7 to 40 hexadecimal characters)	The commit SHA to be reviewed. If this parameter is not specified, the changes in the current working directory are reviewed by default.   Supported commit ID formats:      • Raw SHA (7 to 40 hexadecimal characters): <b>8b480d1</b> or <b>8b480d1a3b5c...</b>    • Colon: <b>commitId:8b480d1</b>    • Equal sign: <b>commitId=8b480d1</b>    • Natural language: <b>Audit 8b480d1</b>, <b>Check 8b480d1</b>, or <b>Review 8b480d1</b>

**Examples:** The following three examples demonstrate how to call **/review** in different ways and what are the differences.

- Reviewing all uncommitted code changes  
`/review`
- Reviewing all code changes in a specified commit  
`/review 8b480d1a3b5c4d6e7f8a9b0c1d2e3f4a5b6c7d8e`
- Reviewing a specified commit using a natural language description  
`/review Review the code changes in commit 8b480d1a3b5c4d6e7f8a9b0c1d2e3f4a5b6c7d8e`

## /codeagent-review-rules

**Function:** Manages code review rule sets, including generating, displaying, and deleting rules.

**Syntax:** [Table 16-4](#) provides four types of syntax for the `/codeagent-review-rules` command and their functions. Select the operation you need and press **Enter** to execute it.

**Table 16-4** /codeagent-review-rules syntax

Syntax	Description
<code>/codeagent-review-rules &lt;File path&gt;&lt;Rule set name&gt;</code>	Generates a new rule set based on a specification file.
<code>/codeagent-review-rules --generate &lt;File path&gt;&lt;Rule set name&gt;</code>	
<code>/codeagent-review-rules --show-rules &lt;Rule set name&gt;</code>	Lists enabled rules ( <b>enabled=true</b> ).
<code>/codeagent-review-rules --delete-rules &lt;Rule set name&gt; &lt;Rule ID_1,Rule ID_2,...&gt;</code>	Disables a specified rule by setting its <b>enabled</b> field to <b>false</b> .

**Table 16-5** /codeagent-review-rules command parameters

Parameter	Description
File path	Local path for storing a specification file. Currently, only specification files in the following formats are supported:     • PDF (.pdf)    • WORD (.docx or .doc)    • Markdown (.md)
Rule set name	Name of a rule set generated based on a specification file. The rule set name facilitates subsequent reference and management.
Rule ID list	ID of the rule to be disabled. Use commas (,) to separate multiple IDs, for example, <b>rule_001,rule_002</b> .



**Examples:** The following examples demonstrate how to use **/codeagent-review-rules** to automate the management of code review rules.

- Generating a rule set based on a PDF specification file  
`/codeagent-review-rules --generate /path/to/Python Coding Style Guide.pdf Python Coding Style Guide V3.0`
- Viewing enabled rules  
`/codeagent-review-rules --show-rules Python Coding Style Guide V3.0`
- Disabling specified rules  
`/codeagent-review-rules --delete-rules Python Coding Style Guide V3.0 RULE_001,RULE_005`

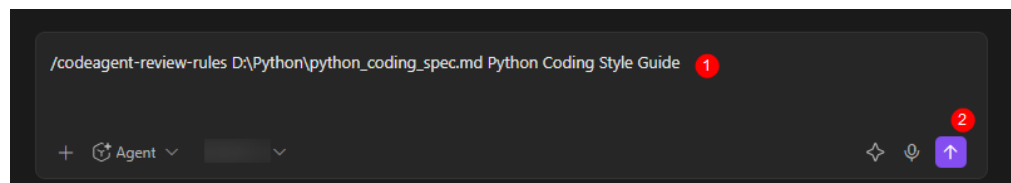
## Using These Commands

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

**Step 2** Run **/codeagent-review-rules** to generate a rule set based on a local specification file.

1. Click  in the chat box, choose **Shortcuts**, and select **/codeagent-review-rules** from the displayed menu.
2. Enter the location of the specification file and the name of the rule set to generate, and click .

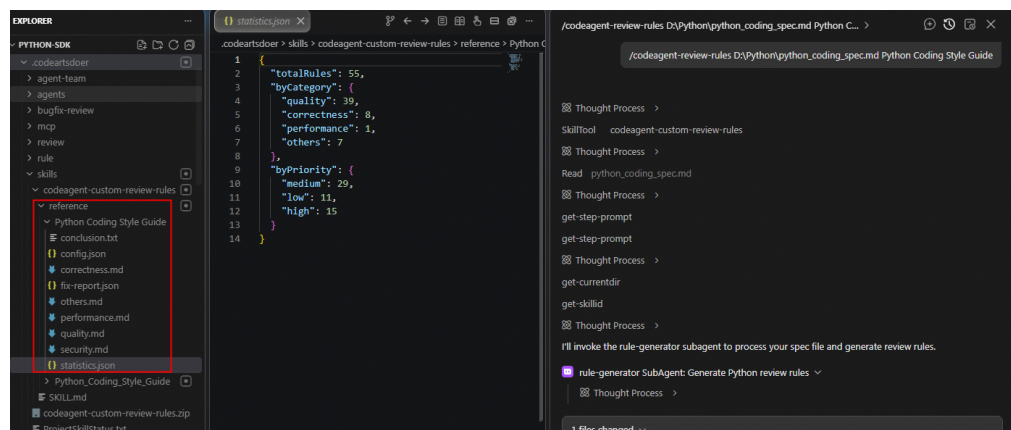
**Figure 16-1** Generating a rule set



3. View the generated rule set.

A rule set folder named **codeagent-custom-review-rules** is generated in the **.codeartsdoer/skills** directory of the explorer's project root directory.

**Figure 16-2** Viewing a generated rule set



**Step 3** Run **/review** to comprehensively review code changes based on the generated rule set.

1. Click  in the chat box and choose **Shortcuts**.

2. Select **/review** from the displayed menu and click .
3. Check the code review result.

In the **.codeartsdoer/review** directory of the explorer's project root directory, a folder named in the format "year-month-day\_hour-minute-second-millisecond\_random string" is generated. Each folder contains two subdirectories: **preprocess** and **report**. **preprocess** stores preprocessed files including code changes. **report** stores the final review report.

----End

## 16.2 Code Fixing

CodeArts Agent IDE automatically fixes issues from various sources, such as manual commits, AI reviews, and tool alarms. It can fix both a single issue and multiple ones. It efficiently finds and resolves issues to significantly improve code quality and development efficiency.

### Constraints

Table 16-6 Constraints

Category	Description
Environment requirements	• Before running code fixing commands, ensure that the code to be fixed matches the JSON file from CodeArts Check.    • The <b>/fix-review-batch</b> command depends on the review agent's review result <b>review.json</b>.    • The <b>/fix-codeartscheck-batch</b> command depends on the check result provided by <a href="#">CodeArts Check</a>.

### Code Fixing Commands

The following describes how to use the four core commands.

#### **/bugfix**

**Function:** Automatically invokes the repair expert agent to fix code issues based on the trouble tickets submitted by users.

**Syntax:**

```
/bugfix <description>
```

**Table 16-7** /bugfix command parameters

Parameter	Description
description	Issue description, for example, "A null pointer exception occurs during user login."

**Example:**

```
/bugfix "A null pointer exception occurs during user login."
```

**/fix-review**

**Function:** Fixes code based on a single review comment from the code review agent. This command processes only one comment at a time and is suitable for fine-tuning.

**Syntax:**

```
/fix-review <review_text>
```

**Table 16-8** /fix-review command parameters

Parameter	Description
review_text	Description of an issue in the JSON file.

**Example:**

```
/fix-review " {
 "severity": "High",
 "ruleId": "G.CLS.06-CPP",
 "ruleDescription": "Do not call virtual functions in constructors or destructors.",
 "file": "src/Base.cpp",
 "line": "4",
 "issueDetails": "A base class constructor calls the virtual function Initialize(). When a derived object is
constructed, the base constructor is called first. At this time, the object's derived part is not yet constructed,
and the virtual function table still points to the base class. Therefore, Initialize() calls Base::Initialize()
instead of Derived::Initialize(), resulting in the failure to implement polymorphic behavior. In addition, the
Derived constructor calls Initialize() again (line 28), so Initialize() is called twice (the base version is called
first, and then the derived version). This is not the expected behavior.",
 "fixSuggestion": "Extract the initialization logic required in the constructor as a non-virtual function (for
example, private InitImpl). Alternatively, directly perform the initialization required by the base class in the
Base constructor instead of calling a virtual function. For example:\n\n// Solution 1: Extract the initialization
logic as a non-virtual private function. \nBase::Base(const std::string& n) : name(n) {\n DoBaseInit(); //
Non-virtual function \n}\n\nvoid Base::DoBaseInit() {\n std::cout << \"Base Initialize for \" << name << \"\\n\";\n}\n\n// Solution 2: Directly perform initialization in each constructor without indirectly calling a
virtual function. \nBase::Base(const std::string& n) : name(n) {\n std::cout << \"Base Initialize for \" <<
name << \"\\n\";\n}\n\nDerived::Derived(const std::string& n, int v) : Base(n), value(v) {\n std::cout <<
\"Derived Initialize for \" << name << \" with value \" << value << \"\\n\";\n}\n}"
```

**/fix-review-batch**

**Function:** Fixes code based on multiple review comments from the code review agent. This command is suitable for resolving all review comments after code is committed in batches, which improves efficiency.

**Syntax:**

```
/fix-review-batch [filter] [file1.json file2.json file3.json ...]
```

**Table 16-9** /fix-review-batch command parameters

Parameter	Description
filter	(Optional) Review comment level, which can be described in natural language, such as major, critical, high level, high risk, medium risk, and low risk.
file1.json file2.json file3.json ...	Path of the JSON file to be used for fixing. You can configure one or more paths. Use spaces to separate multiple paths.   A maximum of 100 review comments are allowed for each JSON file. As shown in <a href="#">Figure 16-3</a>, the total number of review comments for <b>Base.cpp</b> cannot exceed 100 in the JSON file corresponding to <b>Base.cpp</b>.

**Figure 16-3** Review comments for the Base.cpp file

```
Base(const Base&) = delete;\nBase& operator=(const Base&) = delete;\nBase(Base&) = delete;\nBase& operator=(Base&) = delete;\n\nprotected:\n std::string name;\n};\n{\n \"severity\": \"High\", \n \"ruleId\": \"C.LOG.01\", \n \"ruleDescription\": \"Logic error: virtual method called from constructor does not dispatch to derived override\", \n \"file\": \"src/Base.cpp\", \n \"line\": \"4\", \n \"issueDetails\": \"Base::Base() calls Initialize(), which is virtual. During Base construction of a Derived object, the dynamic type is still Base, so Base::Initialize() is called - NOT Derived::Initialize(). This is a well-known C++ pitfall (see Scott Meyers, Effective C++ Item 9). When Derived d(\\\"test\\\", 42) is constructed: (1) Base::Base() runs and calls Base::Initialize() (not Derived's override), printing 'Base Initialize for test'; (2) Derived::Derived() then calls Initialize() again, this time dispatching to Derived::Initialize(). The net result is Initialize() is called twice with the wrong version called first. The Base constructor's call to Initialize() will never reach a derived override.\", \n \"fixSuggestion\": \"Remove the Initialize() call from Base::Base(). If initialization logic is needed, use a non-virtual helper or a separate init() method called after full construction. For example:\\n\\nBase::Base(const std::string& n) : name(n) {\\n // Do not call virtual Initialize() here\\n}\\n\\nThen call Initialize() explicitly after construction, or use a factory function that constructs the object and then calls Initialize().\" \n}, \n{\n \"severity\": \"High\", \n \"ruleId\": \"C.LOG.01\", \n \"ruleDescription\": \"Logic error: Derived::Print() signature mismatch with Base::Print() prevents override\", \n \"file\": \"src/Base.cpp\", \n \"line\": \"24\", \n \"issueDetails\": \"Base::Print() is declared as 'virtual void Print() const;' (with const qualifier), but Derived::Print() is declared as 'void Print() override;' (without const). The signature mismatch means Derived::Print() does NOT override Base::Print(). The 'override' keyword should cause a compilation error here because no matching virtual function exists in the base class. If this compiles (e.g., if the compiler does not enforce override checking), calling Print() through a Base pointer/reference to a Derived object would incorrectly call Base::Print() instead of Derived::Print().\", \n \"fixSuggestion\": \"Add the 'const' qualifier to Derived::Print() to match the base class signature:\\n\\nvoid Print() const override;\\n\\nAnd update the definition on line 35 accordingly:\\n\\nvoid Derived::Print() const { ... }\" \n} \n}
```

**Example:**

- Basic usage: Pointing to one or more JSON files  

```
/fix-review-batch /path/to/review.json
/fix-review-batch /path/to/review1.json /path/to/review2.json
```
- Filter by severity: Fixing only the issues marked as "High severity".  

```
/fix-review-batch High severity /path/to/review.json
```

**/fix-codeartsccheck-batch**

**Function:** Fixes code based on the alarms generated by [CodeArts Check](#).

**Syntax:**

```
/fix-codeartsccheck-batch file_name1.json file_name2.json ...
```

**Table 16-10** /fix-codeartscheck-batch command parameters

Parameter	Description
file_name1.json file_name2.json ...	CodeArts Check scans code and generates a JSON file that records all the issues found in the code. Set this parameter to the path of the JSON file. You can configure one or more paths and separate them with spaces.

**Example:**

```
/fix-codeartscheck-batch D:\TestCode\codecheck-result1.json codecheck-result2.json
```

## Code Fixing Example

The following describes how to use the **/fix-review-batch** command to fix code.

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

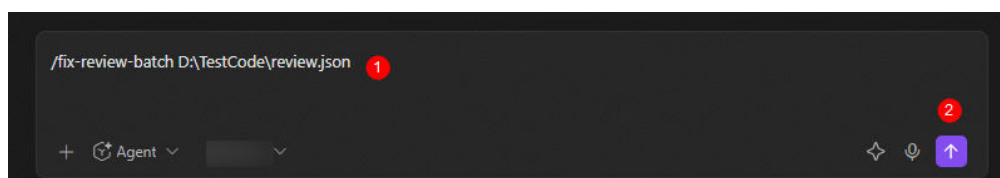
**Step 2** (Optional) Check whether there is a review comment file in JSON format. If yes, skip this step. If no, run the **/review** command to obtain the file.

1. Click  in the chat box and choose **Shortcuts**.
2. Select **/review** from the displayed menu and click .
3. After the command is run, the review comment file **review.json** is automatically generated.

In the **.codeartsdoer/review** directory of the explorer's project root directory, a folder named in the format "year-month-day\_hour-minute-second-millisecond\_random string" is generated. View the **review.json** file in the **report** directory of this folder. For details about the **/review** command, see [Code Review](#).

**Step 3** Run **/fix-review-batch** to resolve the review comments in the **review.json** file.

1. Click  in the chat box, select **Shortcuts**, and choose **/fix-review-batch** from the displayed menu.
2. Enter the path of a JSON file that contains code review comments, and click .

**Figure 16-4** Fixing code

3. Wait until the command is run. The agent will display the number of issues fixed, their severities, and the fix content.

**Step 4** Check whether the issues have been fixed based on the review comments.

Run **/review** again to check whether the issues have been fixed.

**----End**

# 17 Settings

## 17.1 IDE Tool Panel

Use the CodeArts Agent extension integrated in CodeArts Agent IDE to make your development and programming more intelligent and easy.

**Table 17-1** Functions description for the tool panel of CodeArts Agent IDE

Tool	Description	Typical Scenario
Explorer	It is a file center of CodeArts Agent IDE, which helps you manage workspace files in a clear tree structure.     - You can create, rename, delete, and drag and drop files for efficient file management.   - You can experience automatic project recognition and display optimization. For example, Java projects are automatically displayed in the package view, where the <b>DEPENDENCIES</b> part lists the dependencies of the Maven or Gradle projects. This greatly simplifies the development process.	When developing a new module, you can right-click the panel to create folders and class files. To view the external dependency libraries used in the project, expand <b>DEPENDENCIES</b>.

Tool	Description	Typical Scenario
Search	CodeArts Agent IDE provides a powerful global search engine to help you quickly find what you need. Press <b>Ctrl+Shift+F</b> to search. The following search methods are supported:      • Text search: Quickly find specified text across all files within a project.    • File search: Quickly locate files within a project.    • Symbols of code elements such as classes, methods, and variables	When renaming a method, you can search for all references to the method to make sure the integrity of the modification.
Source Control	CodeArts Agent IDE is deeply integrated with the Git version control system and provides an easy-to-use graphical user interface (GUI). You can view the file change status (modified, new, or untracked) in one click, compare code differences, stage, submit, push, and pull code, and efficiently resolve merge conflicts. This out-of-the-box tool frees you from memorizing complex commands. The learning costs are low and the version management is more intuitive and efficient.	This is a tool you will use many times every day. After modifying a piece of code, you can view the changes, fill in the commit message, and push the changes to the shared code repository of the team.
Run and Debug	CodeArts Agent IDE provides you with a complete and integrated program execution and debugging environment. You can configure the debugging environment for multiple languages such as Python, Java, and Node.js with just one click. During debugging, advanced functions such as breakpoint setting, step-by-step execution (step-in/step-out), real-time variable monitoring, and call stack viewing are supported, helping you monitor the running of each line of code and improving development efficiency and accuracy.	When a program logic error is detected, you can configure the debugger in this panel, set breakpoints on suspicious code lines, start debugging to trace the program execution process and status changes step by step, and accurately locate the error.



Tool	Description	Typical Scenario
Extensions	It works as CodeArts Agent IDE's extension market where you can discover, add, and manage diverse extensions to CodeArts Agent IDE for your growing development requirements, such as adding languages, creating custom code check rules, changing themes, and integrating debugging tools.   You can find and use high-quality extensions from official and community sources compatible with CodeArts Agent IDE, including the Huawei Cloud extension market and Open VSX, to build personalized development environments.   In addition, CodeArts Agent IDE features Huawei's self-developed Java language support to deliver a next-generation development experience. It not only provides deep adaptation for mainstream frameworks but also boasts native AI-enhanced features. You can enable them directly by searching for and installing the CodeArts AI Features Extension.	When developing a Java project, you can search for and install the CodeArts AI Features Extension to get syntax highlighting, code generation, and more.
Remote Development	CodeArts Agent IDE supports remote development, allowing you to bypass geographical limitations and efficiently write, debug, and deploy code on the cloud. For details, see <a href="#">Remote Development</a>.	When facing tasks that require compiling large-scale Java projects, training AI models, or processing massive amounts of data, your local computer's CPU and memory often fall short. Remote development allows you to directly call high-performance instances on the cloud, making the development process smooth while relieving the burden on your local machine.

Tool	Description	Typical Scenario
Problems	It displays all code issues in a project, including errors, warnings, and informational messages. For example, syntax errors reported by compilers, type mismatches reported by language servers, and style violations reported by code check tools. You can view and locate issues with one click to fix potential defects quickly and continuously improve code quality and development reliability.	After compiling or saving a project, if there are multiple issues, you can open this panel to view all issues and double-click the issue to quickly jump to the corresponding code line for fixing.
Output	It is a general log panel displaying the backend run logs of CodeArts Agent IDE, such as extension installation, build process, and language server output. It displays key information in real time, helping you efficiently track task status and quickly locate issues, improving debugging and troubleshooting efficiency.	If a custom build task fails, you can find its detailed error logs here and then locate the fault.
Debug Console	After a debugging session is started, this console displays the output information of the program running in real time, including the output of statements such as console.log, print, and System.out.println. The console integrates the Read-Eval-Print Loop (REPL, interactive interpretation environment), which allows you to view the current value in real time by entering a variable name or expression, facilitating debugging and logic verification and improving development efficiency.	When debugging a Node.js application, you can manually call a function and pass test parameters to the function in the console to view the return value.
Terminal	CodeArts Agent IDE integrates system terminals (such as PowerShell and Bash) and automatically switches to the root directory of the current project. You can run project-related commands without leaving the editor, improving efficiency and operation smoothness.	You can run project build commands (such as npm run build), install Python packages (pip install), and run Git commands (git status) and other operations that need to be completed through the command line.

Tool	Description	Typical Scenario
Port	The port window shows and manages network port details. It lets developers check which ports are in use and check their connection statuses. Developers can also troubleshoot, inspect, and control network communications through this window to keep the app's networking features working smoothly.	During multi-task joint commissioning, the port window shows the real-time status of multiple services like the frontend, backend, and database. It lets you check service health instantly, identify issues, and restart faulty services promptly.
Git Graph	It graphically displays the branch structure, merge history, and commit records of the entire Git repository. Compared with the traditional git log command, this method is more intuitive and can clearly display the branch evolution, merge relationships, and associations between commits.	When you take over a project with a long history or the branch merging policy of the team is complex, you can open the Git graph to quickly understand the development process of the project and the integration path of each function.
Notification	When the CodeArts Agent client is minimized or loses focus, both successful and failed task executions will automatically trigger audio alerts and message notifications. By enabling audio and message notifications, you can stay informed of task completion statuses and respond quickly to anomalies without having to constantly stare at the screen, significantly improving multitasking efficiency.	For long-running, high-concurrency, and unattended AI task scenarios, where task failures can halt progress and exceptions cannot afford to be missed, it is highly recommended to enable both notifications and audio alerts to ensure you are immediately aware of the task status.

## 17.2 Global Settings of the IDE

Click  in the upper right corner of CodeArts Agent IDE to open the settings page. Then you can customize the global settings based on your preferences and habits.

### Account

View information about the current login account.

- If you are logged out, click **Log In** to access the login page.

- If you are logged in, the name and ID of your current account are displayed. Click **Log Out** to log out of the current account.

General

Figure 17-1 General

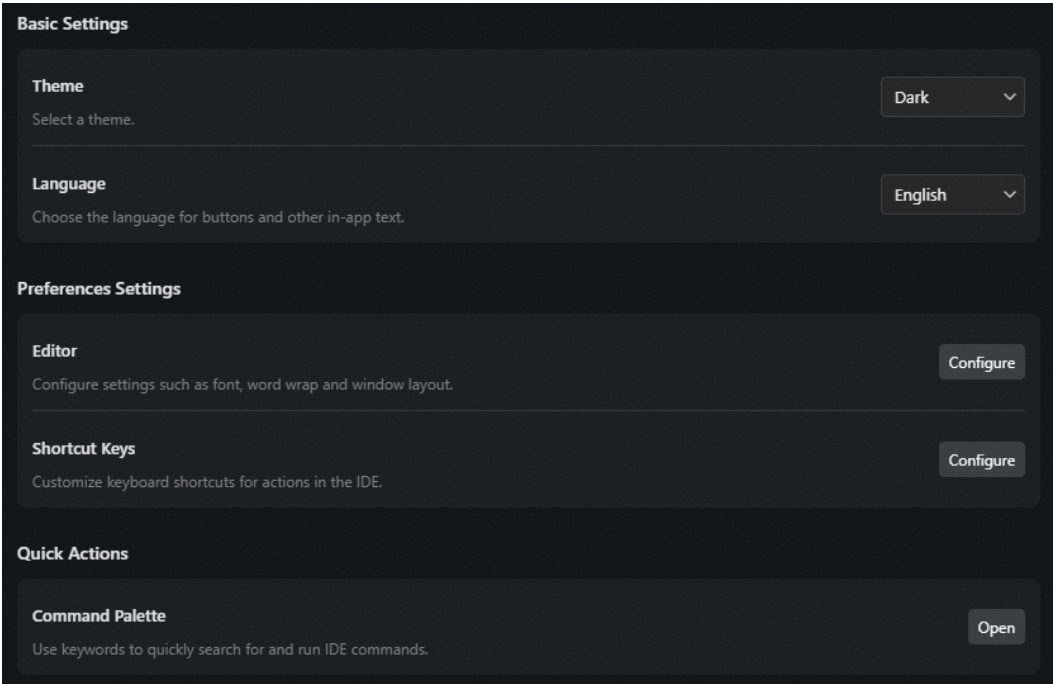


Table 17-2 General configuration items

Type	Configuration Item	Description
Basic settings	Theme	Choose the visual style for the CodeArts Agent IDE interface.  Currently, dark and light themes are supported, and you can also choose <b>More</b> to install other themes from the marketplace. Switch freely based on your environment and personal preference to improve coding comfort and visual experience.
	Language	Choose the language for button labels and other text. Switch between Simplified Chinese and English as needed.
Preference settings	Editor	Set editor parameters.  Click <b>Configure</b> to open the editor settings page and adjust editor features like fonts, workspace, and windows for a personalized development setup.

Type	Configuration Item	Description
	Shortcut Keys	Customize shortcut keys. Click <b>Configure</b> to customize shortcuts for CodeArts Agent IDE tasks and boost your productivity.
Quick actions	Command Palette	Click <b>Open</b> to access the command palette. Type keywords to find and run commands easily in CodeArts Agent IDE.

Chats

Figure 17-2 Display of gutter icons

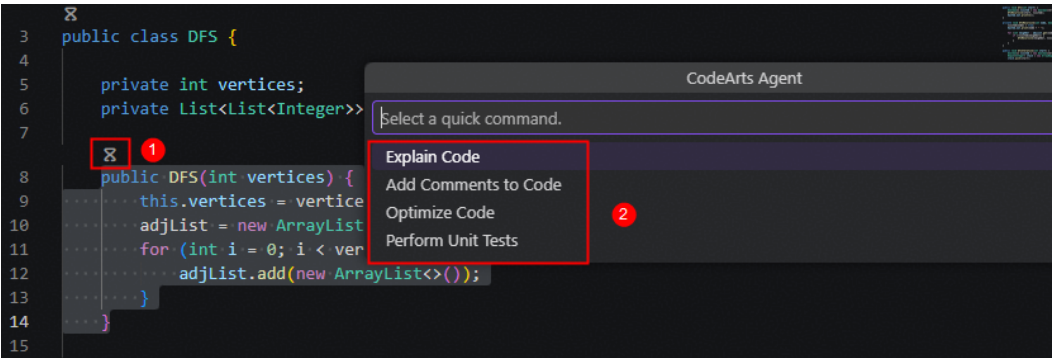


Table 17-3 Configuration items

Type	Configuration Item	Description
Chat settings	Response Language	Determine the response language for the model during chats. Choose a specific language or let the model adapt to the user's input language automatically, ensuring flexibility across various situations. Available choices include:     ● <b>Auto</b>: The model detects the question's language and responds accordingly.    ● <b>Chinese</b>: The model responds exclusively in Chinese, no matter what language the question uses.    ● <b>English</b>: The model responds exclusively in English, no matter what language the question uses.

Type	Configuration Item	Description
	Send Shortcut	<b>Press Enter to send a message, Ctrl+Enter to start a new line:</b> After entering a prompt in the input box, press <b>Enter</b> to send it. If you want to wrap the line and continue typing, press <b>Ctrl+Enter</b> .
		<b>Press Ctrl+Enter to send a message, Enter to start a new line:</b> After entering a prompt in the input box, press <b>Ctrl+Enter</b> to send it. If you want to wrap the line and continue typing, press <b>Enter</b> .
	Voice Settings	Customize the language displayed after speech-to-text conversion. Select one of the following options from the drop-down list:     • <b>Chinese:</b> Regardless of whether the voice input is in Chinese, English, or a mix of both, the system converts it into the corresponding language (Chinese, English, or mixed of both).    • <b>English:</b> Regardless of whether the voice input is in Chinese, English, or a mix of both, the system automatically converts it into English.
Basic settings	Font Size	Set the font size on the CodeArts Agent interface.
	Gutter Icons	Gutter icons, as shown in <a href="#">Figure 17-2</a> , are shortcut icons displayed next to definition lines of functions, classes, and other elements in the gutter area of the code editor. They are used to quickly perform common development operations.  Select the desired features (such as code explanation, commenting, optimization, and unit testing) from the drop-down list. The selected features will be displayed in the gutter area next to the corresponding code line. Click the icon and select the corresponding function to quickly perform operations on functions, classes, and other elements, improving development efficiency.

Type	Configuration Item	Description
Advanced settings	Minimum Log Level	Choose the minimum log level needed to maintain both system efficiency and effective troubleshooting.      • <b>Configuration principles:</b> You are advised to set the log level to <b>Debug</b>. Debug logs record internal code execution details during development and debugging, making it easier to locate and troubleshoot problems that occur during code execution.    • <b>Log level:</b> Log levels increase in severity like this: <b>Debug</b>, Info, <b>Warn</b>, and <b>Error</b>.    – Debug: records detailed internal information during runtime, such as request details and process steps. This level is mainly used for debugging and fault locating in the development phase.    – Info: records key status information during normal runtime, such as service startup and request processing summaries, to help you understand the basic system workflow.    – Warn: records potential issues that may affect system stability, such as missing configurations and insufficient resources. These issues do not directly cause program interruption.    – Error: records errors that have caused some functions to be abnormal or operations to fail, such as API calling failures and data parsing errors. These errors require timely attention and rectification.

Type	Configuration Item	Description
Agents	Terminal Command Running Mode	Select a policy for the agent to execute terminal commands based on your security requirements. By default, all terminal commands must be manually approved by you before execution.      • <b>Running in Sandbox:</b> Operations are performed automatically in a secure sandbox by default.    • <b>Auto Running:</b> The agent directly executes all commands without your approval. <b>For security purposes, you are advised to select Auto Running only when necessary. In this mode, the agent bypasses all security checks and may perform high-risk operations without notifications.</b>    • <b>Manual Running:</b> Before executing any command, the agent sends a confirmation prompt to you. You need to manually confirm it.
	Terminal Command Running Mode > Command Whitelist	Add the prefix of a specific command to the whitelist as required. Commands added to the whitelist bypass the sandbox mechanism and are executed outside the sandbox.   <b>NOTE</b> This configuration item is displayed only when <b>Terminal Command Running Mode</b> is set to <b>Running in Sandbox</b>.
	Terminal Command Running Mode > Custom Sandbox Settings	By configuring the <b>sandbox.json</b> file, you can customize files and the network access scopes for processes within the current project's sandbox environment.   <b>NOTE</b> This configuration item is displayed only when <b>Terminal Command Running Mode</b> is set to <b>Running in Sandbox</b>.



Type	Configuration Item	Description
	Terminal Command Running Mode > Network Access	The agent provides three network access policies for you to choose from. Configure access control rules based on actual service requirements and security levels.      • <b>Network-wide:</b> Access to all internal and external network resources is allowed.    • <b>Local:</b> Only the local network (LAN/ intranet) can be accessed. Access to the Internet is not allowed.    • <b>Blocked:</b> All network connections are blocked, and access to internal or external resources is prohibited.    • <b>Custom:</b> Allows you to modify the JSON policy configuration file to customize the file and network access scope for processes within the project sandbox environment.      <b>NOTE</b> This configuration item is displayed only when <b>Terminal Command Running Mode</b> is set to <b>Running in Sandbox</b>.
	Auto-approve > Edit	Allow the agent to call tools like edit, write, and deleteFile to edit files on your computer.
	Auto-approve > Browser Access	Allow the agent to access websites in the browser.
	Auto-approve > Web Crawler	Allow the agent to access and capture specified web page content.
Agent security policy	Dotfile Protection	When enabled, the agent is not allowed to modify configuration files starting with a dot, such as <b>.env</b> and <b>.gitignore</b> .
	Edit Scope Protection	When enabled, the agent can only add, modify, and delete files in the current project directory. It cannot edit files in other directories.
	Read Scope Protection	When enabled, the agent can only access and read files in the current project directory. It cannot access files in other directories.
Intelligent Q&A	Suggested Questions	When enabled, the system predicts three relevant questions using your preferences and conversation history after answering your question. Click these suggestions to ask them instantly. For details, see <a href="#">Suggested Questions</a> .

## Agents

Agents are programming assistants tailored for diverse development scenarios. CodeArts Agent not only provides various built-in agents but also supports custom agent capabilities. By configuring prompts, skills, and tool sets within a custom agent as required, you can create a dedicated assistant to handle complex development tasks more efficiently.

- **Enterprise agents:** available to all enterprise users, created by enterprise administrators, and stored in the cloud.
- **Team agents:** restricted to designated teams, created by enterprise or team administrators, and stored in the cloud.
- **Project agents:** effective only for the current project, distributed along with the repository, and stored locally.
- **User agents:** comprised of **local** and **cloud** agents.
  - Local user agents: take effect for all projects under the current user and are stored locally.
  - Cloud user agents: are available only to the creator and are stored on the cloud.
- **Built-in system agents:** The system provides multiple out-of-the-box agents that require no manual creation, aiming to provide efficient programming assistance for various development scenarios.

For details about built-in agents and custom agents, see [Built-in Agents](#) and [Custom Agents](#), respectively.

## Models

In addition to system built-in models, enterprise members can also connect third-party large language models to CodeArts Agent IDE. If you have purchased the CodeArts Agent professional edition, you can also add third-party models on the console.

For details about models, see [Models](#).

Code Generation

Figure 17-3 Code generation

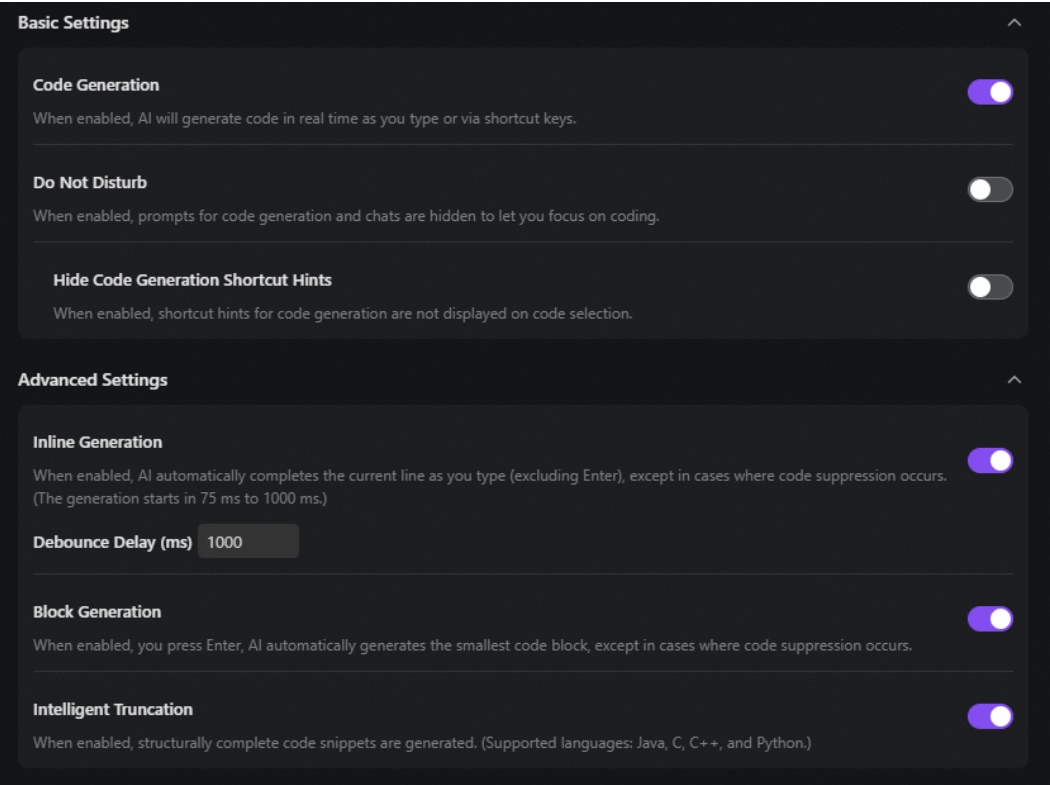


Table 17-4 Code generation settings

Type	Configuration Item	Description
Basic settings	Code Generation	When enabled, CodeArts Agent provides code generation features, such as real-time suggestions while typing and full code generation via keyboard shortcuts.
	Do Not Disturb	When enabled, code generation and chat prompts will be hidden to help you focus on coding. Specify this parameter based on your requirements.
	Hide Code Generation Shortcut Hints	Shortcut tips that appear during code selection will not be displayed if this option is toggled on.

Type	Configuration Item	Description
Advanced settings	Inline Generation	When enabled, CodeArts Agent instantly generates code for the current line as you type code (before hitting <b>Enter</b>). These suggestions appear within 75 to 1,000 ms.   You can also adjust the debounce delay to prevent accidental triggers. The debounce delay (ms) sets the waiting time before triggering inline generation (ranging from 75 to 1,000 ms).   <b>NOTE</b> The tool stops suggesting code if the system faces heavy loads or encounters complex code contexts.
	Block Generation	When enabled, CodeArts Agent will suggest ways to complete the smallest code block as soon as you hit <b>Enter</b>.   <b>NOTE</b> The tool stops generating code blocks if the system faces heavy loads or encounters complex code contexts.
	Intelligent Truncation	When enabled, CodeArts Agent detects the code context and generates full code blocks automatically.   Java, C, C++, and Python are currently supported.

MCP

CodeArts Agent connects to an MCP server using the Model Context Protocol (MCP). It leverages the server's extra tools and resources to enhance its capabilities. You can add an MCP server here. For details, see [MCP](#).

Skills and Rules

- You can view and configure skills and rules for enterprises, teams, projects, and users. This ensures consistent yet adaptable workflows tailored to general standards or individual preferences.
- Enterprise skills and rules: Standardized skill libraries and business rules that are centrally built, managed, and distributed at the company or large organization level. The core objective is to eliminate execution deviations and ensure that different teams and individuals strictly adhere to unified operation standards, security protocols, and process guidelines in complex business scenarios. This enhances overall operational consistency and controllability.

- Team skills and rules: Skills and rules that are uniformly configured for teams or departments to standardize team operations and ensure consistent operations among all members. Only team administrators or department administrators can create, modify, and delete these skills and rules. Common members can only use them.
- Project skills and rules: Skills and rules that apply only to the current project. Bound to the project repository, they are distributed alongside it to address specific project needs.
- User skills and rules are classified into local and cloud ones.
  - Local ones: effective for all projects of the current user, formulated based on personal habits or preferences, and stored locally.
  - Cloud ones: formulated based on personal habits or preferences, available only to the members themselves, and stored in the cloud.
- Built-in skills and rules: basic skills preset in the system, globally available, and ready to use out of the box. They provide stable, continuous baseline functionality and can be used directly without any additional configuration.
- Skill marketplace: a hub of multiple ready-to-use skills designed to help rapidly boost development efficiency.

For details about how to configure skills and rules, see [Skills](#) and [Rules](#), respectively.

## Memory

Memory allows CodeArts Agent to adapt to how you work. The agent will record your habits and preferences, such as your preferred answer style and language.

For details, see [Memory](#).

## Repository Index

The repository index function is designed to enhance code retrieval efficiency. CodeArts Agent supports building both local user indexes and cloud shared indexes, allowing flexible adaptation based on your actual needs to achieve efficient information retrieval and management.

For details about repository indexes, see [Repository Index](#).

## Lab Features

Language Server Protocol (LSP) is a standardized protocol for providing language services. By decoupling language services from the IDE, LSP enables an independent language server process to deliver core capabilities such as code analysis, error diagnosis, fast navigation, and code refactoring.

CodeArts Agent now features a built-in local LSP function. When enabled, the agent can leverage definitions and symbol data from language services for more precise code comprehension and modification.

For details, see [LSP](#).

## Knowledge Base

CodeArts Agent enhances chat capabilities by utilizing knowledge bases as context. Developers can upload internal documents and technical resources to improve response accuracy.

For details, see [Knowledge Bases](#).

## About CodeArts Agent

**Table 17-5** Configuration items

Type	Configuration Item	Description
About CodeArts Agent	Check for Updates	After you click <b>Check for Updates</b> , the system automatically checks the current version. Meanwhile, the update status appears in the upper right corner of CodeArts Agent IDE. Hovering over it allows you to view the upgrade progress, new features, and error alerts in real time, keeping you informed throughout the upgrade process for a smoother experience.
	Product Homepage	Visit the official website of CodeArts Agent and get the latest installation package.
	Documents	Read the CodeArts Agent documents for quickstart and usage tips.
	What's New	Access the latest updates of CodeArts Agent.
	FAQs	Access frequently-asked questions to quickly get guidance and solve problems when using CodeArts Agent.
	Knowledge Base Mgmt	Access the knowledge base module on the CodeArts Agent console to handle and use all your knowledge resources efficiently.
	Privacy Statement	Visit the privacy statement page of CodeArts Agent to review its terms and guidelines.
	Feedback	Open the feedback page to share issues or suggestions. Your input helps us enhance the service.

Type	Configuration Item	Description
Version Updates	Update Preferences	When enabled, the system will automatically download and install updates whenever a new version is detected.     • <b>Check every 12 hours:</b> The system will automatically check for updates every 12 hours while running.    • <b>Check on each startup:</b> The system will only perform a version check automatically when the IDE starts up, without repeated checks during runtime.
	Show Release Notes After Update	When enabled, a window will automatically pop up after each update is completed to display the release notes for the current update.

## 17.3 Process Resources Explorer

The process resources explorer monitors CodeArts Agent IDE's resource use in real time, helping you adjust processes as needed and maintain its stability and performance.

### Opening the Process Resources Explorer

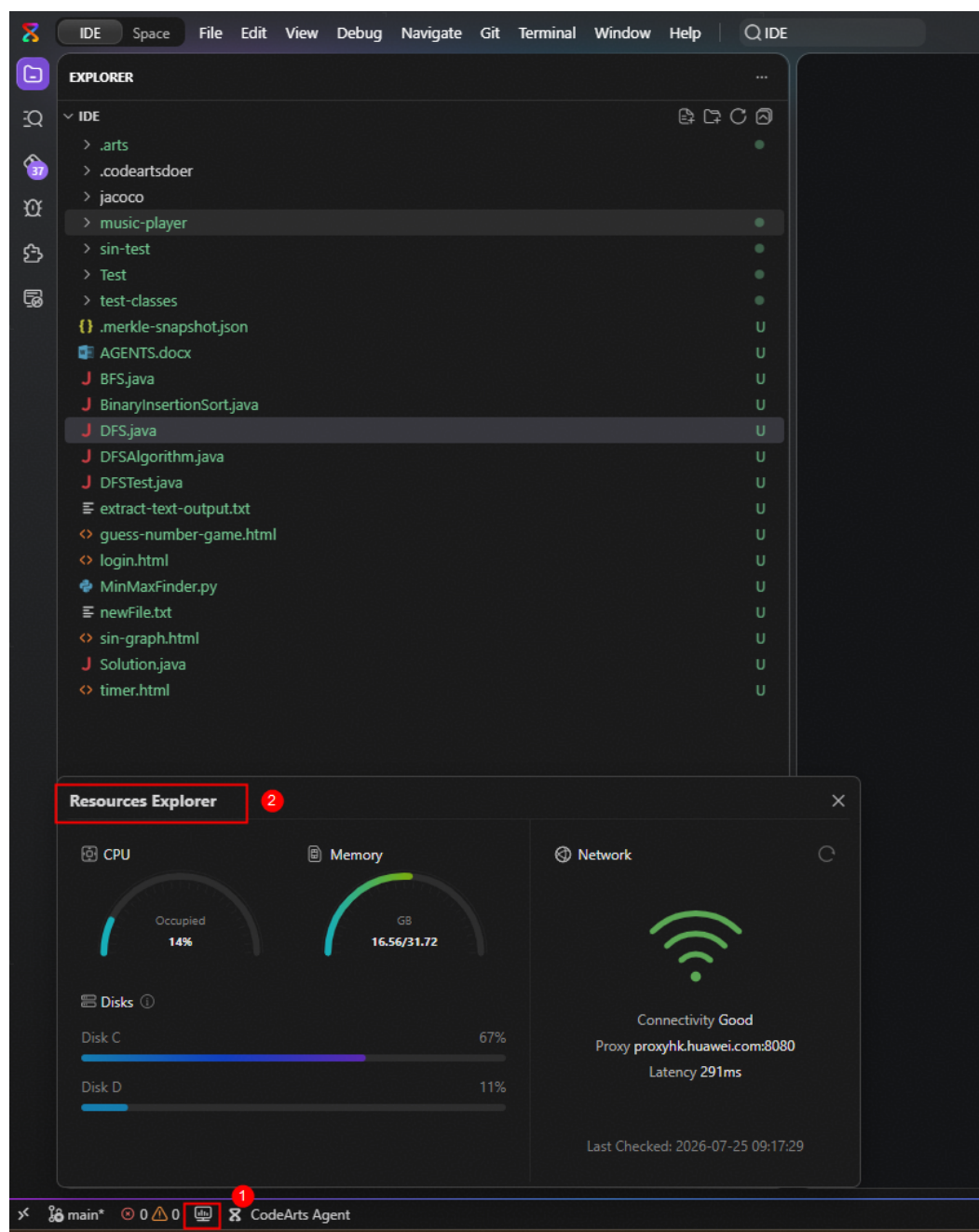
In the lower left corner of CodeArts Agent IDE, click  (normal state) to open the resources explorer.

#### NOTE

If the process resources explorer icon is displayed in another state, the meanings are as follows:

- : Network latency is too high.
- : Network connection is interrupted.
- : Memory/CPU utilization is too high.



**Figure 17-4** Opening the process resources explorer

## Viewing Resource Usage

The process resources explorer can:

- Monitor the CPU usage of the current system.
- Monitor the memory usage of the current system.
- Display the disk usage of the current system.
- Display the read/write speed, usage, and I/O status in real time when you hover the mouse over the disk information icon, giving you quick insights into storage performance.



- Monitor network connectivity, proxy configurations, and latency in real time. This allows you to quickly determine whether an abnormal response stems from a network fault, helping pinpoint the exact root cause of the issue.

## 17.4 Network Proxy

If you are using an enterprise network, you might see a prompt indicating that the model failed to be loaded and cannot be used. If this problem occurs, you can configure a network proxy by referring to this section, restart the tool, and then use CodeArts Agent IDE.

### Setting a Network Proxy

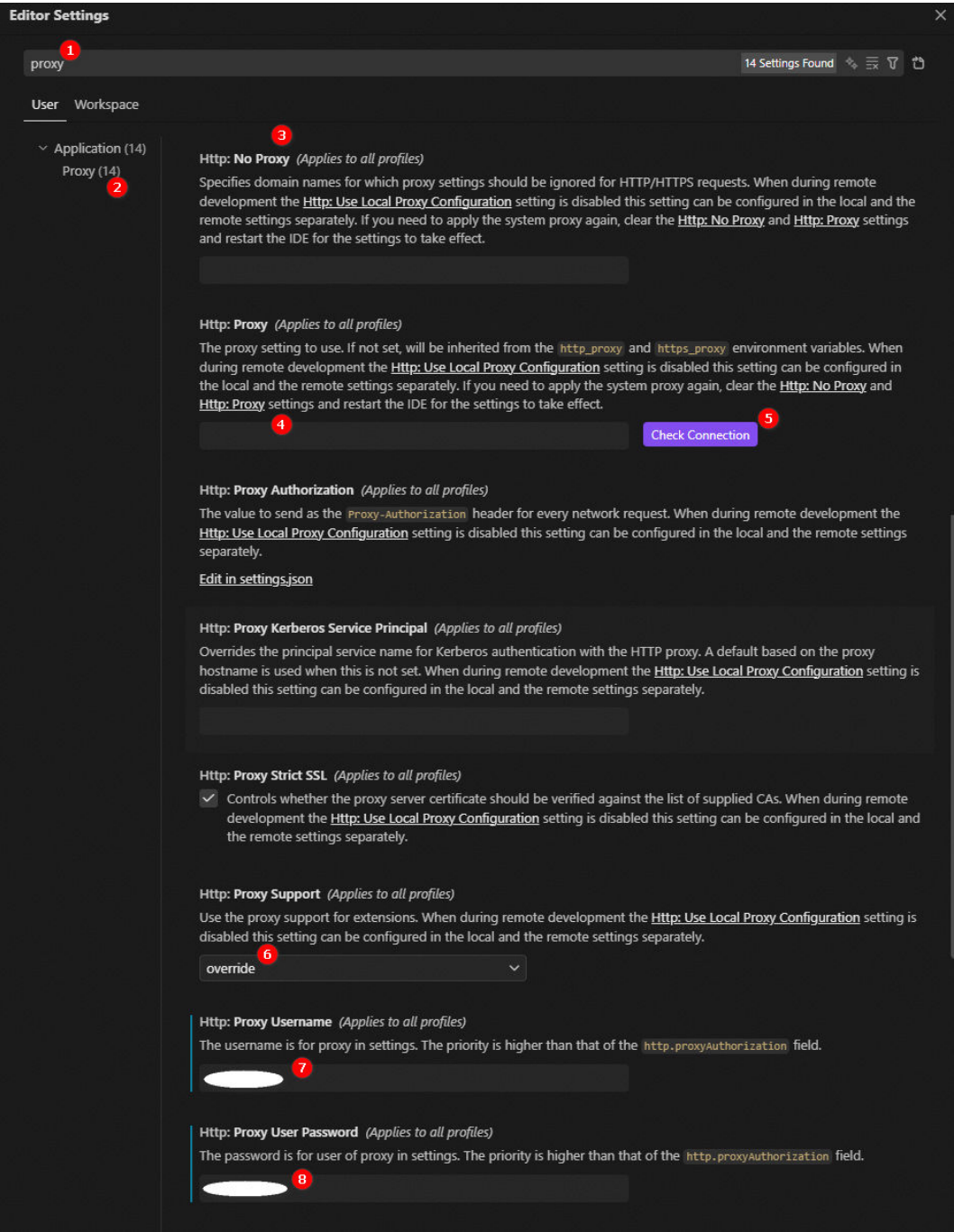
**Step 1** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.

**Step 2** Choose **General > Preference Settings**, and click **Configure** next to **Editor**.

**Step 3** On the editor's **Settings** page, search for **Proxy**, click **Proxy Server**, and enter the proxy information as shown in the following figure.

Proceed with subsequent operations after the connection is successful according to step 5 in the figure below.

Figure 17-5 Setting the proxy



**CAUTION**

If your **Proxy User Password** contains special characters such as @ or %, you must escape them before entering the password. Otherwise, the proxy configuration will be invalid.

-----End

# 18 Obtaining Logs

---

To ensure normal system operation, improve troubleshooting efficiency, and locate and analyze problems in a timely manner, you can obtain and view the log information of CodeArts Agent by following the instructions provided in this section.

## Log Level Description

Log levels increase in severity like this: Debug, Info, Warn, and Error.

- **Debug:** records detailed internal information during runtime, such as request details and process steps. This level is mainly used for debugging and fault locating in the development phase.
- **Info:** records key status information during normal runtime, such as service startup and request processing summaries, to help you understand the basic system workflow.
- **Warn:** records potential issues that may affect system stability, such as missing configurations and insufficient resources. These issues do not directly cause program interruption.
- **Error:** records errors that have caused some functions to be abnormal or operations to fail, such as API calling failures and data parsing errors. These errors require timely attention and rectification.

## Setting the Log Level

Set the log level based on the actual scenario to control the output content and details of logs, thereby improving log readability and system debugging efficiency. You are advised to set the log level to Debug. Debug logs record internal code execution details during development and debugging, making it easier to locate and troubleshoot problems that occur during code execution.

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

**Step 2** Click  in the upper right corner of CodeArts Agent IDE to open the settings page.

**Step 3** In the navigation pane, choose **Chats**.

**Step 4** In the **Advanced Settings** area, select **Debug** from the **Minimum Log Level** drop-down list.

----End

## Obtaining Logs

In CodeArts Agent, logs are classified into the following types. Obtain logs based on the actual requirements.

**The following operations apply only to Windows and macOS.**

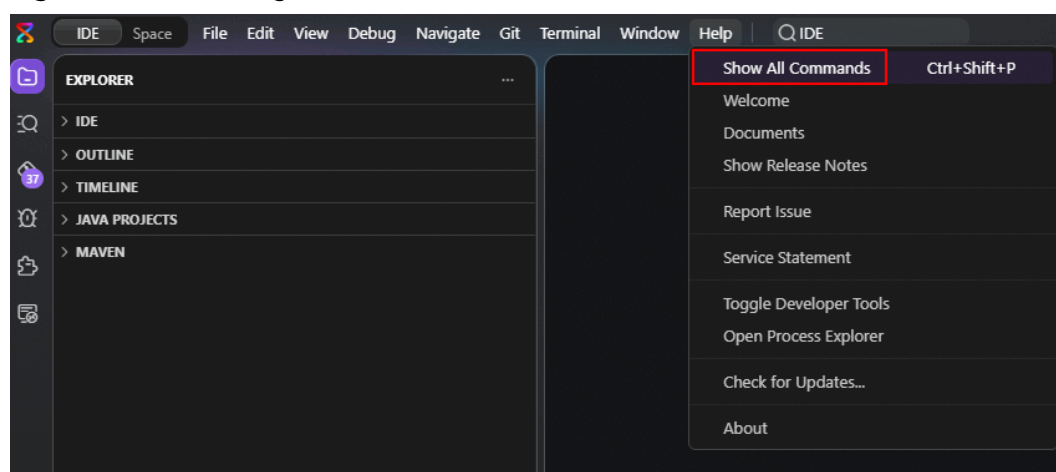
### Obtaining Agent Logs

If CodeArts Agent encounters issues such as dialog box anomalies, setting errors, task failures, garbled output, or frozen statuses, you can follow the steps below to obtain agent logs for troubleshooting.

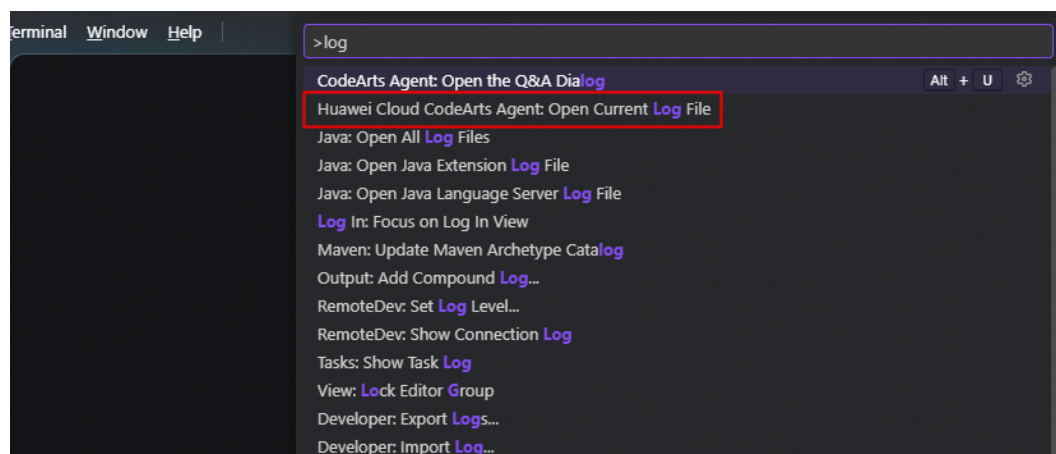
- **Obtaining Session Logs**

**Step 1** Open CodeArts Agent IDE and choose **Help > Show All Commands** from the top menu bar.

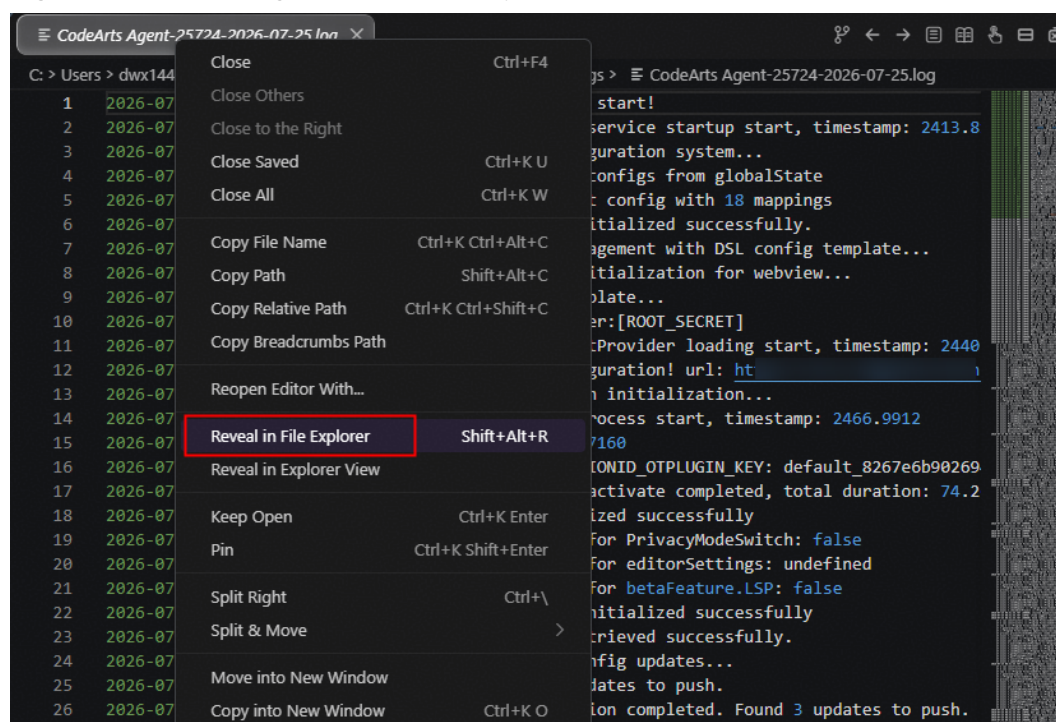
**Figure 18-1** Choosing Show All Commands



**Step 2** Enter **log** to search for the command, and select **Huawei Cloud CodeArts Agent: Open Current Log File**. The log content is displayed in the editor.

**Figure 18-2** Selecting Huawei Cloud CodeArts Agent: Open Current Log File

**Step 3** Right-click the log name at the top of the editor and choose **Reveal in File Explorer** from the shortcut menu. The folder where the log is stored is displayed.

**Figure 18-3** Choosing Reveal in File Explorer

**Step 4** Obtain the agent session logs of the required date.

----End

- **Obtaining Kernel Logs**

View kernel logs in the local `%USERPROFILE%\.codeartsdoer/codearts-data/log` directory.

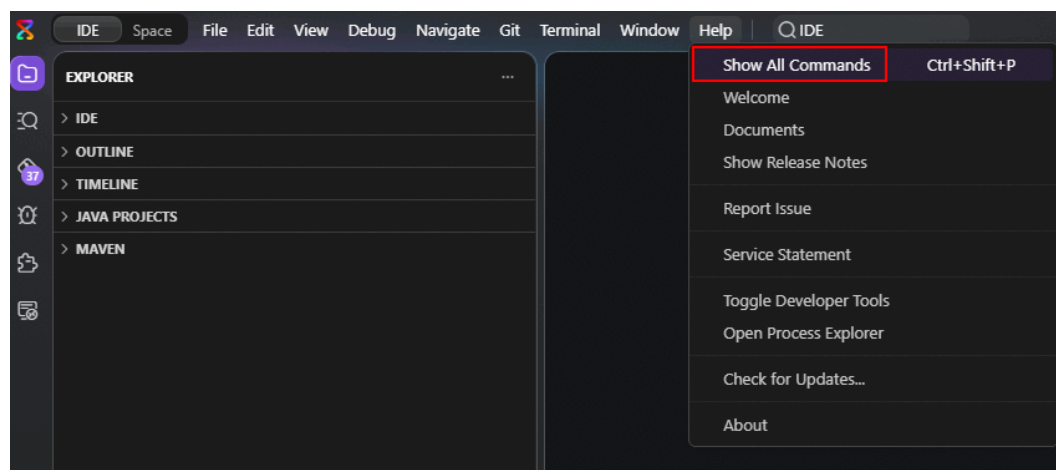
**Figure 18-4** Obtaining kernel logs

agent-kernel-sdk-2026-07-15T143401-0.log	2026/7/15 14:57
agent-kernel-sdk-2026-07-15T161411-0.log	2026/7/16 14:24
agent-kernel-sdk-2026-07-16T142426-0.log	2026/7/16 14:56
agent-kernel-sdk-2026-07-16T145627-0.log	2026/7/16 14:57
agent-kernel-sdk-2026-07-16T145743-0.log	2026/7/16 15:12
agent-kernel-sdk-2026-07-16T152821-0.log	2026/7/16 17:11
agent-kernel-sdk-2026-07-16T171207-0.log	2026/7/17 9:37
agent-kernel-sdk-2026-07-17T094308-0.log	2026/7/17 9:44
agent-kernel-sdk-2026-07-17T094441-0.log	2026/7/17 9:51
agent-kernel-sdk-2026-07-17T095236-0.log	2026/7/17 9:53
agent-kernel-sdk-2026-07-17T095413-0.log	2026/7/17 9:55
agent-kernel-sdk-2026-07-17T095603-0.log	2026/7/17 9:56
agent-kernel-sdk-2026-07-17T095624-0.log	2026/7/17 16:47

## Obtaining IDE Logs

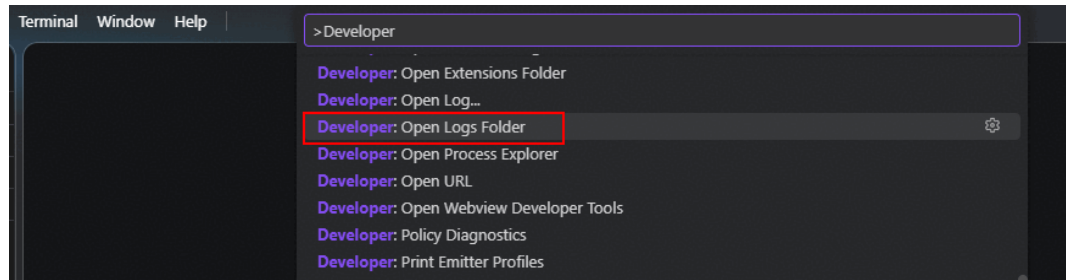
If the IDE encounters functional anomalies related to directory display, code editing, menu bar configurations, or login, check the IDE logs to troubleshoot and locate the issue.

- Step 1** Open CodeArts Agent IDE and choose **Help > Show All Commands** from the top menu bar.

**Figure 18-5** Choosing Show All Commands

- Step 2** Enter **Developer** to search for commands, and select **Developer: Open Logs Folder**.

**Figure 18-6** Selecting Developer: Open Logs Folder



**Step 3** Obtain **main.log**.

-----End

### Obtaining Logs on IDE Process Exceptions

If the IDE process exits unexpectedly or encounters an out of memory (OOM) error, you can collect the IDE crash logs to help troubleshoot the fault.

How to obtain: Navigate to the local path **%USERPROFILE%/AppData/Roaming/codearts-agent/Crashpad/reports** to view the IDE process exception logs.



# 19 Feedback

CodeArts Agent allows you to quickly report problems and suggestions encountered during use, helping the product team continuously optimize the product. It also effectively connects developers to promote communication and cooperation.

## Submitting Feedback in the IDE (Recommended)

If you use CodeArts Agent IDE, you can submit feedback in the IDE directly.

**Step 1** Log in to CodeArts Agent IDE as instructed in [Quick Start](#).

**Step 2** In the upper right corner, click  and choose **Feedback**.

You can also click  in the upper right corner of CodeArts Agent IDE to open the settings page. Then on the **About CodeArts Agent** tab, click **Feedback**.

**Step 3** Select the feedback type.

- Submitting bugs: Feedback on issues discovered in the software, including interface errors, function exceptions, and operation freezing.
- My suggestions: This is used to collect your innovative ideas or suggestions on product functions. Describe your suggestions as detailed as possible. We will take your suggestions and consider them in the development plan later.

**Step 4** Enter your feedback by referring to [Table 19-1](#).

**Table 19-1** Feedback

Parameter	Description
Description	Explain the issue or provide suggestions clearly. Keep it no more than 1,000 characters. For feedback related to chats on CodeArts Agent, copy and paste the relevant chat details here.
Screenshot	In addition to describing the issue in text, you can also upload a screenshot of the issue.  You can upload up to 10 images at a time. The image format must be JPG, PNG, or GIF, and the total size cannot exceed 5 MB.



Parameter	Description
Contact	Provide your contact information to help us better locate issues.
Auxiliary information	Choose whether to provide plug-in information and application logs as required. The information is used only to help locate, reproduce, and resolve issues.      • Plug-in information: names, versions, and memory usage of all enabled plug-ins.    • Application logs: logs generated when CodeArts Agent is used.

**Step 5** Click **Submit**. After the feedback is submitted successfully, a success message is displayed.

----End

## Submitting a Service Ticket

If you encounter any issues when using CodeArts Agent, you can also [submit a service ticket](#) to Huawei Cloud for customer service support.