CodeArts Agent

User Guide (Extensions)

Issue	01
Date	2026-07-30

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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 the CodeArts Agent Extension?

The CodeArts Agent extension can be installed in two mainstream programming environments: Visual Studio Code and JetBrains. It delivers core capabilities such as code generation, code completion, R&D knowledge Q&A, and unit test case generation, significantly enhancing developers' R&D efficiency and providing a high-quality intelligent coding experience.

Application Scenarios

- **Rapid development of new projects**

  You can get started and deliver projects faster by quickly generating the project framework, service code, and prototype based on the CodeArts Agent capabilities.

- **Developer-friendly code maintenance**

  CodeArts Agent can parse the project code repository and identify semantics and service logic. It accelerates onboarding for legacy systems and streamlines ongoing maintenance and iteration.

- **Smart coding and creative development**

  CodeArts Agent helps with code writing, completion, and generation across all scenarios, adapting to multiple tech stacks and significantly reducing the repetitive coding work.

- **Code quality control**

  Both the explore and standard modes balance development flexibility and compliance requirements. Built-in coding specifications and security monitoring ensure code security, compliance, and maintainability.

- **Debugging and bug fixes**

  CodeArts Agent can intelligently analyze logs and code stacks to locate root cause issues. It will then provide a solution and automatically generate code to rectify the fault.

- **Collaborative creation and retention of knowledge assets**

  CodeArts Agent helps accumulate project knowledge and unify coding standards, which accelerates employee onboarding and improves R&D collaboration efficiency.

Function Description

The table below lists the main functions of the CodeArts Agent extension available for Visual Studio Code and JetBrains.

Table 1-1 Function description

Function	Description
Agent	The agent is used to understand developer needs and plans tasks automatically. It enables full process automation from requirement analysis to coding, boosting efficiency in end-to-end development. For more information, see **Agent Chats**.
Code Generation	CodeArts Agent can generate context-based code suggestions in the editor. **Code generation** can be triggered either **automatically** or through **shortcut keys**. For more information, see **Code Generation**.
Intelligent Q&A	CodeArts Agent delivers end-to-end support for coding, testing, and release. It acts as a conversational sidekick that keeps your team in flow by bringing the right project knowledge when you need it. As the conversation continues, it maintains multi-turn context, cutting time-to-answer. For more information, see **Intelligent Q&A**.
Unit Test	The Unit Test is an end-to-end test case generation agent that provides core capabilities throughout the entire process, including test design, test case generation, and intelligent test case repair. For more information, see **Unit Test (Agent)**.
Adding Context	CodeArts Agent introduces context information to improve Q&A accuracy. Context helps the agent understand the background and intent of your question better to generate more practical and valuable answers. For more information, see **Adding Context**.
Multimodal Input	• Visual Studio Code allows you to upload images or attachments or enable voice input to integrate them into the chat context. • Mainstream JetBrains IDEs support only image uploads. You can upload 1 to 5 images per request, with a maximum total size of 2 MB. Supported formats include PNG, JPG, JPEG, and WebP. For more information, see **Multimodal Input**.

Function	Description
Repository Index	CodeArts Agent parses and indexes code to help you quickly search for and retrieve related information from code repositories. For more information, see **Repository Index**.
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. For more information, see **MCP**.
Skills	Professional knowledge can be encapsulated into skills. You can select the most appropriate skills according to specific development scenarios and collaboration needs, achieving flexible and efficient code management. For more information, see **Skills**.
Chat History	You can view your chat history with the AI, including previous questions, answers, and context. For more information, see **Chat History**.
Hooks	With hooks, you can insert custom logic into key lifecycle nodes of the CodeArts Agent IDE and extension to extend functionality without modifying any existing code. For more information, see **Hooks**.
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. For more information, see **Slash Commands**.
Obtaining Logs	You can obtain and view log information to improve troubleshooting efficiency and locate and analyze problems in a timely manner. For more information, see **Obtaining Logs**.

2 Quick Start

2.1 Visual Studio Code

Constraints

The CodeArts Agent extension for Visual Studio Code is only available for Windows, with Windows 11 (x64) recommended. If you are using Windows 10 (x64), ensure that the Visual Studio Code version is 1.95.3 (released on November 13, 2024) or later. You are advised to upgrade Visual Studio Code to the latest stable version.

Preparations

- You have downloaded and installed the latest version of **Visual Studio Code** (1.69 to 1.103).
- Before coding with CodeArts Agent, **create a HUAWEI ID**.
  a. Visit the **Huawei Cloud official website** and click **Sign Up** in the upper right corner.
  b. **Sign up for a HUAWEI ID and enable Huawei Cloud services**.
- If you are using an **IAM user**, use the enterprise administrator account to assign a seat to the user.

Installing the CodeArts Agent Extension

Step 1 Install the CodeArts Agent extension.

1. Open Visual Studio Code and click  in the left navigation pane.
2. In the search box, enter **CodeArts Agent** and press **Enter**.
3. Click **Install**. Visual Studio Code will then download and install the CodeArts Agent extension.

Step 2 Log in to CodeArts Agent.

1. In the left pane of Visual Studio Code, click the CodeArts Agent icon  .

2. Click **Switch to International Site**.
3. Click **HUAWEI ID Login**. The Huawei Cloud login page is displayed.
4. Enter your HUAWEI ID and password and click **LOG IN**.

   If you use CodeArts Agent for the first time, the activation page is displayed. Agree to the service statement and click **Activate Now**. Wait until the package is enabled.

   Wait until the login is successful and then return to Visual Studio Code.

Step 3 Open the folder for storing project files.

If this is your first login, click **Open Folder** in the **Startup** or **Recent** area of the welcome page to open the local folder. In the displayed dialog box, click **Yes, I trust the authors** to trust the folder and enable all functions.

Step 4 (Optional) Log out of CodeArts Agent.

To log out, click  in the chat panel of CodeArts Agent and choose **Sign Out**.

----End

2.2 JetBrains

CodeArts Agent deeply integrates with mainstream JetBrains IDEs, including **PyCharm**, **IntelliJ IDEA**, **WebStorm**, and **CLion**, providing developers with a seamless cloud-based programming experience.

This section uses IntelliJ IDEA as an example to describe how to install CodeArts Agent on the JetBrains platform.

Constraints

The CodeArts Agent extension for JetBrains is only available for Windows, with Windows 11 (x64) recommended. If you are using Windows 10 (x64), ensure that the IntelliJ IDEA version is 2019 or later. You are advised to upgrade IntelliJ IDEA to the latest stable version.

Preparations

- You have downloaded and installed **IntelliJ IDEA** (versions 2022.3 to 2026.1).
- Before coding with CodeArts Agent, **create a HUAWEI ID**.
  a. Visit the **Huawei Cloud official website** and click **Sign Up** in the upper right corner.
  b. **Sign up for a HUAWEI ID and enable Huawei Cloud services**.
- If you are using an **IAM user**, use the enterprise administrator account to assign a seat to the user.

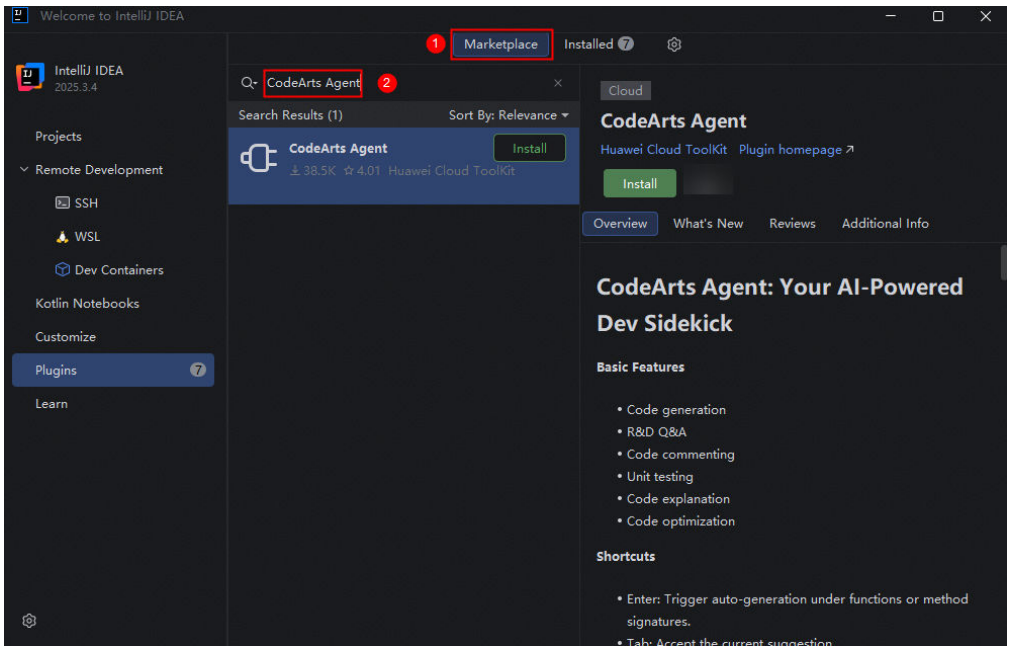
Installing the CodeArts Agent Extension

Step 1 Install the CodeArts Agent extension.

Install the CodeArts Agent extension from the IntelliJ IDEA marketplace:

1. Open IntelliJ IDEA and click **Plugins** in the navigation pane on the left. The **Plugins** page is displayed.
2. Click **Marketplace** on the top of the page and enter **CodeArts Agent** in the search box to search for the CodeArts Agent extension.

Figure 2-1 Searching for CodeArts Agent

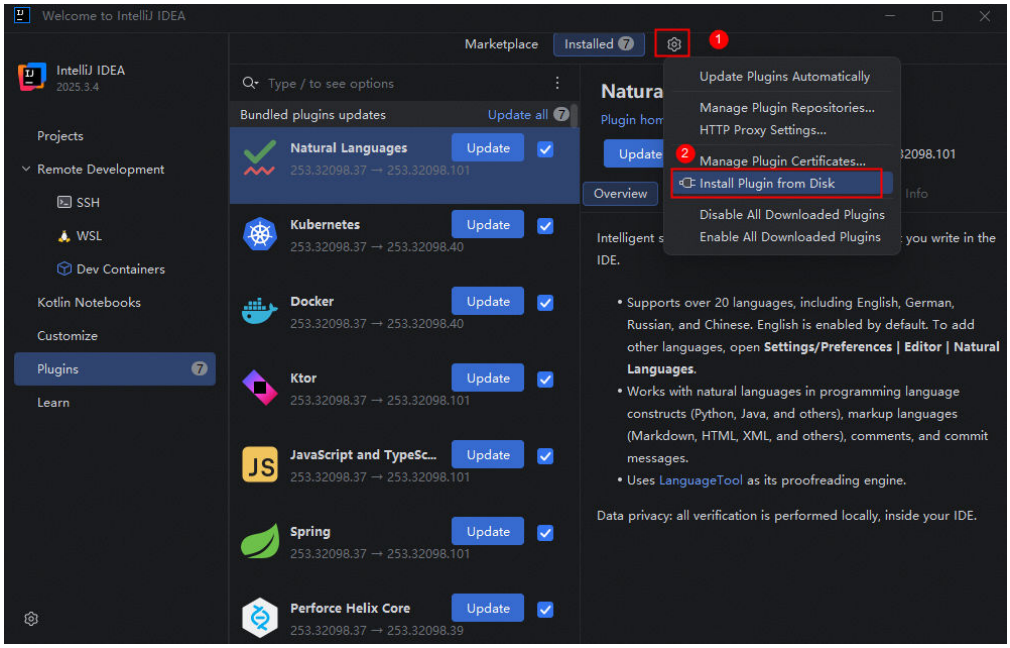


3. Click **Install**. IntelliJ IDEA will then download and install the extension.
4. After the installation is complete, click **Restart IDE** or manually restart IDEA to activate the extension.

You can also download the CodeArts Agent installation package to your computer for offline setup.

1. Go to the CodeArts Agent download page and download the offline installation package based on your IDEA version.
2. Open IntelliJ IDEA and click **Plugins** in the navigation pane on the left. The **Plugins** page is displayed.
3. Click  at the top of the page and select **Install Plugin from Disk**.

Figure 2-2 Selecting Install Plugin from Disk



4. Select the downloaded CodeArts Agent package and click **OK**. IntelliJ IDEA will check the dependencies and install the extension.
5. After the installation is complete, click **Restart IDE** or manually restart IDEA to activate the extension.

Step 2 Create or open a project.

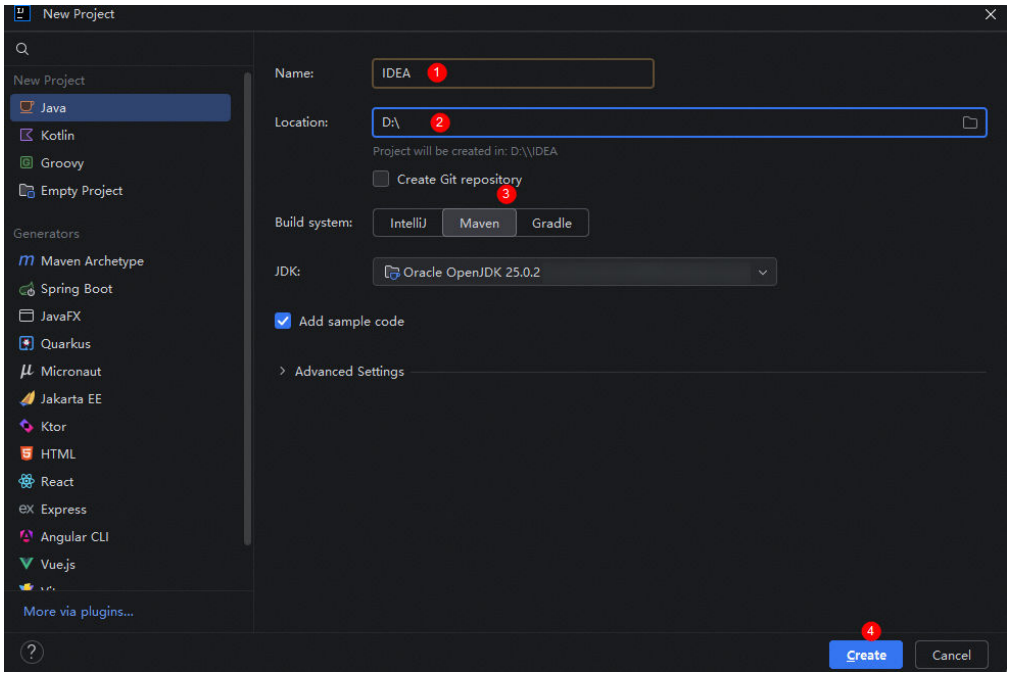
Creating a project: Create a project from scratch and customize your development environment.

1. On the **Welcome to IntelliJ IDEA** page, click **New Project**. The **New Project** dialog box is displayed.
2. Enter the project name, select a path for storing the project, and click **Create**.

CAUTION

Unit tests depend on the data read from **pom.xml**. To ensure unit tests can run properly, set **Build system** to **Maven** when creating a project. After the project is created, a **pom.xml** file is automatically generated in the corresponding folder of the explorer.

Figure 2-3 Creating a project



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

1. On the **Welcome to IntelliJ IDEA** page, click **Open**.
2. Select a local folder to open the project in IntelliJ IDEA.

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

1. On the **Welcome to IntelliJ IDEA** page, click **Clone Repository**.
2. Enter the repository URL, select a local path for storing the repository, and click **Clone**.

Step 3 Log in to CodeArts Agent.

1. On the sidebar of IntelliJ IDEA, click the CodeArts Agent extension icon .
2. Click **Switch to International Site**.
3. Click **HUAWEI ID Login**. The Huawei Cloud login page is displayed.
4. Enter your HUAWEI ID and password and click **LOG IN**.

   If you use CodeArts Agent for the first time, the activation page is displayed. Agree to the service statement and click **Activate Now**. Wait until the package is enabled.

   Wait until the login is successful and then return to IntelliJ IDEA.

Step 4 (Optional) Log out.

To log out, click in the chat panel of CodeArts Agent and choose **Sign Out**.

----End

Related Operations

- If line wrapping does not work in the history preview window, see **What Do I Do If the Chat History Cannot Auto-Wrap?**
- If entering non-English characters in the chat panel causes an error, see **How Do I Fix the Exception Reported When Typing Non-English Characters in the Chat Box?**
- If your JetBrains IDE version is 25.2 and you cannot obtain terminal output, see **What Do I Do When JetBrains IDE Version 25.2 Cannot Display Terminal Output?**

3 Unit Test (Agent)

The Unit Test (Agent) is an end-to-end test case generation agent that provides core capabilities throughout the entire process, including test design, test case generation, and intelligent test case repair. This function can accurately generate test cases that comply with the code logic, automatically identify compilation errors of generated test cases, and intelligently repair test cases with compilation errors, effectively improving the development efficiency and quality of test cases.

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. For more information, see **Built-in Agents**.

Table 3-1 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. It can be invoked using either of the following methods: • **Commands**: Enter a command directly into the chat box to invoke it. For example, use developer-test-agent to generate a unit test for a *specific* method in a *specific* file. • **Right-click shortcut**: This method is only available for IntelliJ IDEA and Java-based unit tests. In agent mode, right-click a **Java method name** and choose **CodeArts Agent** > **Generate Unit Tests (Agent)** from the shortcut menu to quickly enable it.

Agent-based Unit Test Generation

Step 1 Log in to CodeArts Agent as instructed in **JetBrains**.

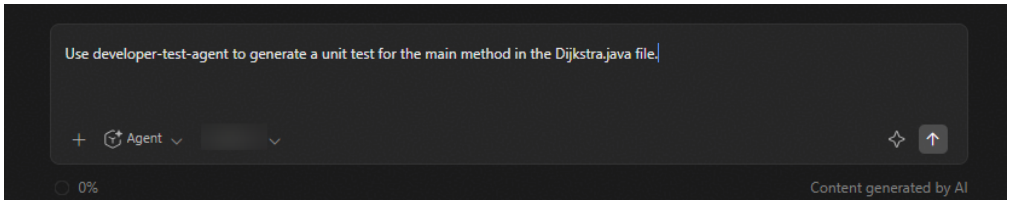
Step 2 On the IntelliJ IDEA sidebar, click the CodeArts Agent icon to open the chat panel.

Step 3 In the chat box, change the model to **Agent**. The selected model is displayed on the right. You can select a model as required from the drop-down list.

Step 4 Select a code snippet and send a unit test generation command to CodeArts Agent. It can be invoked using either of the following methods:

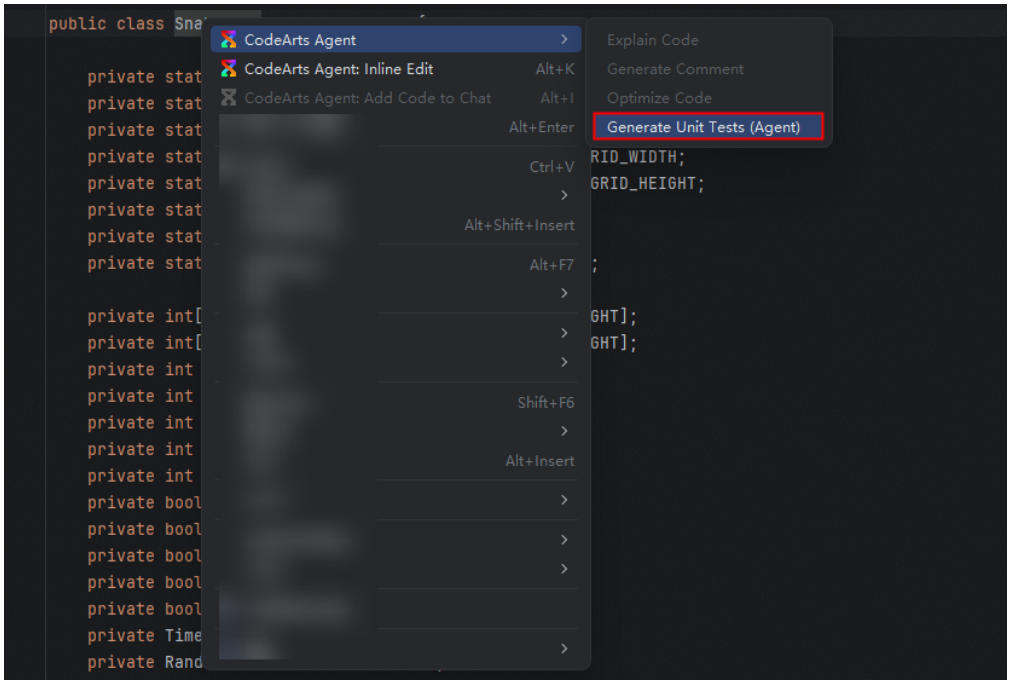
- **Prompts**: Enter a prompt directly into the chat box to invoke it, for example, "Use developer-test-agent to generate a unit test for the main method in the Dijkstra.java file."

  Figure 3-1 Unit test

- **Right-click shortcut**: This method is only available for IntelliJ IDEA of the JetBrains series. In agent mode, right-click a **Java method name** and choose **CodeArts Agent** > **Generate Unit Tests (Agent)** from the shortcut menu to quickly enable it.

Figure 3-2 Choosing Generate Unit Tests (Agent)



Step 5 Check whether a unit test file is generated. If yes, the execution is successful and the test is passed.

----End

4 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 as instructed in **Quick Start**.

Step 2 Open the CodeArts Agent chat panel.

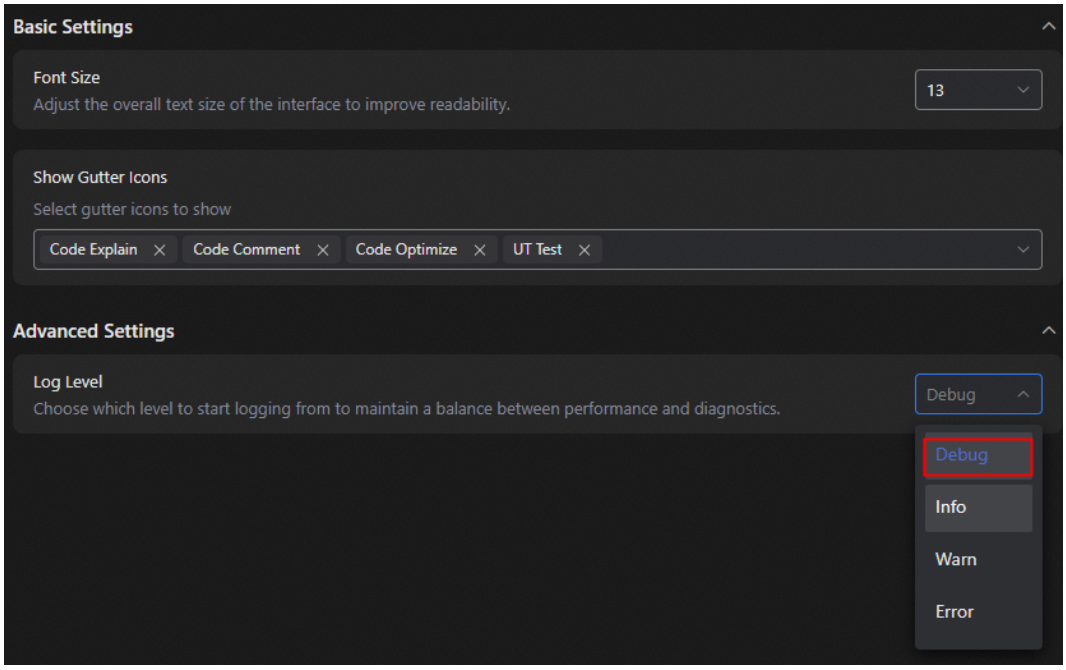
- Visual Studio Code: Click  in the Visual Studio Code sidebar to open the CodeArts Agent chat panel.

- JetBrains (with IntelliJ IDEA as an example): Click in the IntelliJ IDEA sidebar to open the CodeArts Agent chat panel.

Step 3 On the chat page, click in the upper right corner to access the settings page.

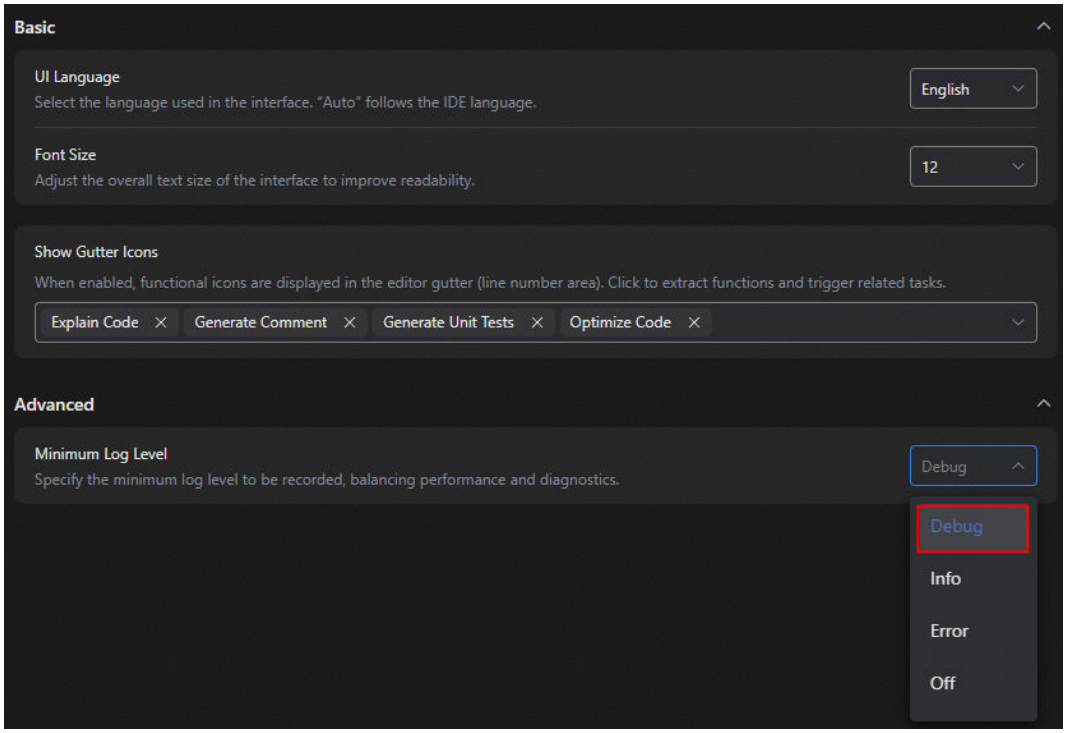
Step 4 Visual Studio Code: In the **Advanced Settings** > **Log Level** drop-down list, select **Debug**.

Figure 4-1 Setting the log level (Visual Studio Code)



Step 5 IntelliJ IDEA: In the **Advanced** > **Minimum Log Level** drop-down list, select **Debug**.

Figure 4-2 Setting the log level (IntelliJ IDEA)



----End

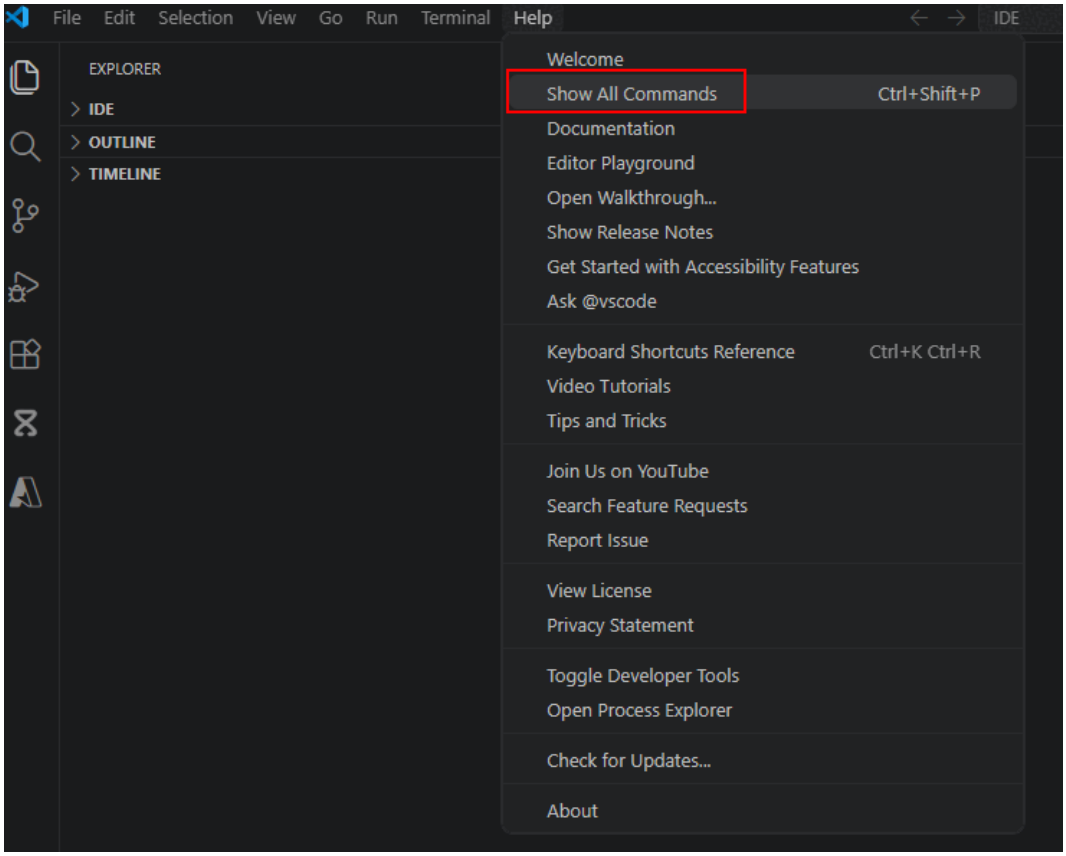
Obtaining CodeArts Agent Logs

The following describes how to obtain logs.

Visual Studio Code

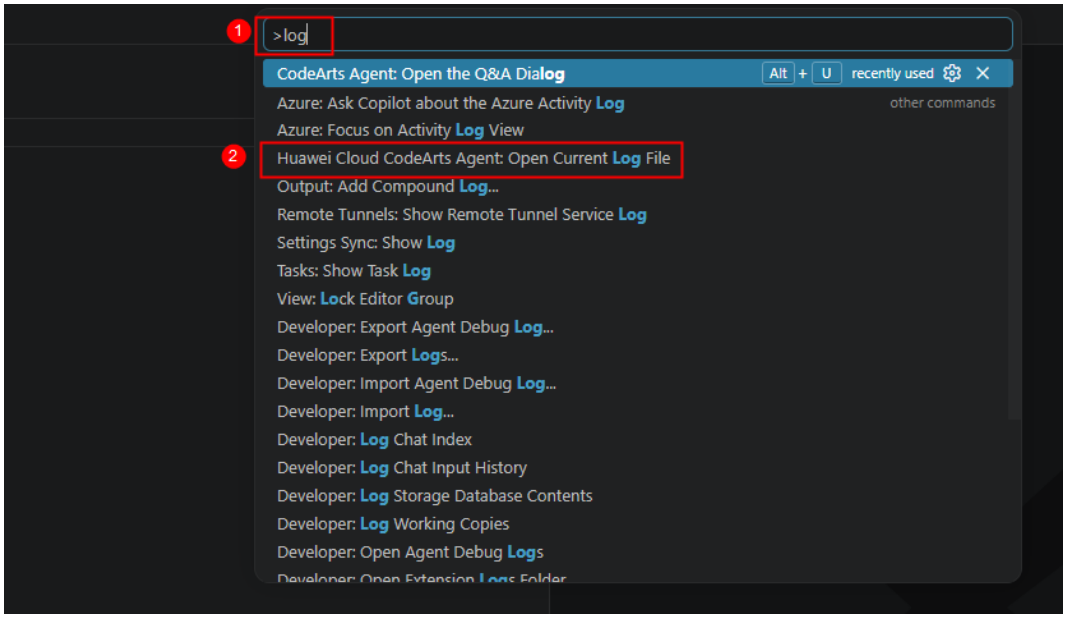
Step 1 Open Visual Studio Code and choose **Help** > **Show All Commands** from the top menu bar.

Figure 4-3 Choosing Show All Commands



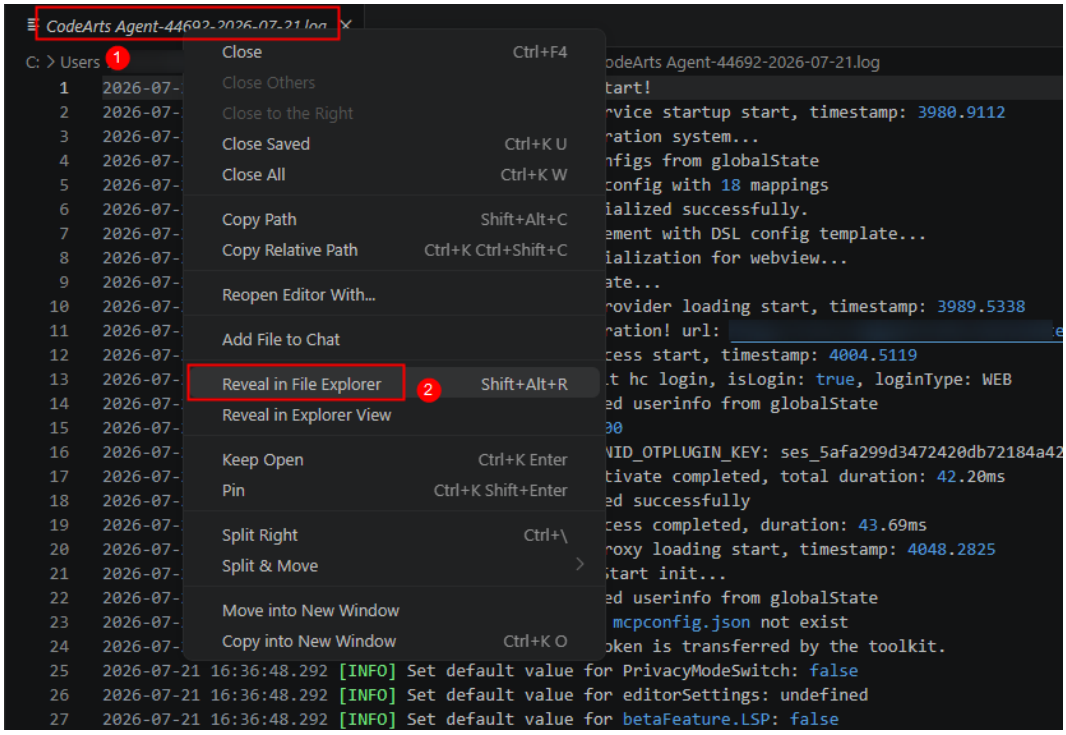
Step 2 Enter **log** to search for the command, and select **Huawei Cloud CodeArts Agent: Open Current Log File**. The log content is displayed in the editor.

Figure 4-4 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 4-5 Choosing Reveal in File Explorer



Step 4 Obtain the agent process log of the corresponding date as required.

Step 5 Obtain kernel logs.

View kernel logs in the local **%USERPROFILE%/.codeartsdoer/vscode-data/log** directory.

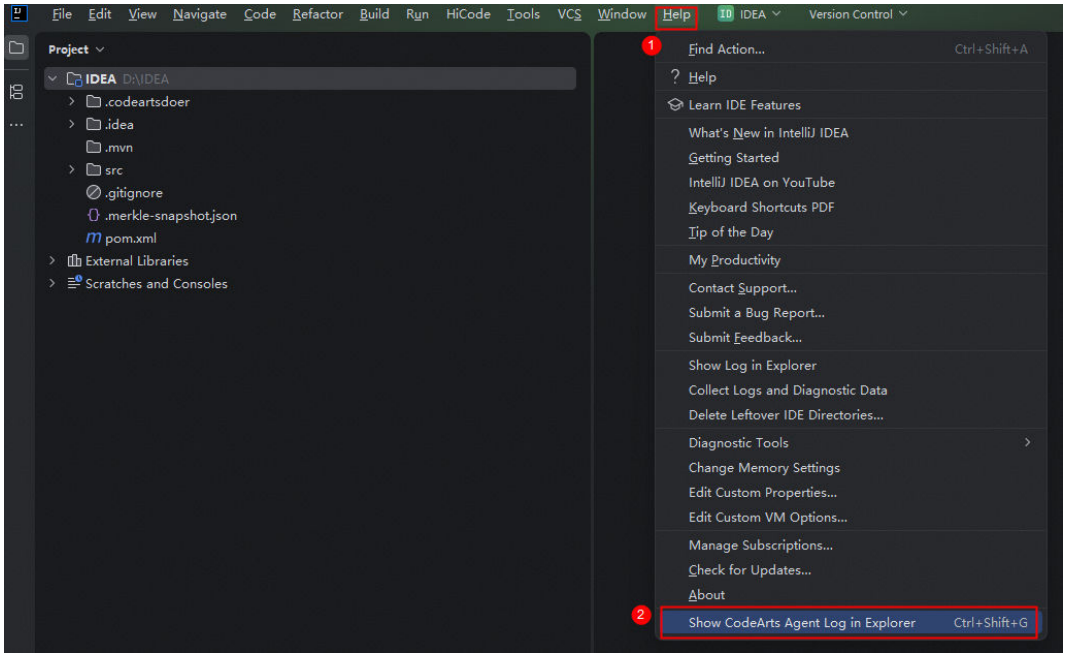
----End

JetBrains

The following uses IntelliJ IDEA as an example to describe how to obtain CodeArts Agent logs.

Step 1 Open IntelliJ IDEA, click in the upper left corner, and choose **Help** > **Show CodeArts Agent Log in Explorer**. The folder where the logs are stored is displayed.

Figure 4-6 Choosing Show CodeArts Agent Log in Explorer



Step 2 Obtain the agent log of the corresponding date as required.

For the latest log, no date identifier is added to the end of the log name.

Step 3 Obtain kernel logs.

View the kernel logs in the local **%USERPROFILE%/mylogs** directory.

----End

5 Settings

5.1 Basic Settings

Click  in the upper right corner of the CodeArts Agent chat panel to open the settings page.

NOTE

The configuration items vary depending on the tool. This section uses Visual Studio Code as an example.

General Settings

Table 5-1 General settings

Item	Description
UI Language	Select the display language. If set to **Auto**, it will automatically adapt to the language settings of IntelliJ IDEA. **NOTE** This configuration item is displayed only in JetBrains tools (PyCharm, IntelliJ IDEA, WebStorm, and CLion).
Font Size	Set the font size of the CodeArts Agent page.

Item	Description
Show Gutter Icons	Gutter icons 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.
Minimum Log Level	Choose the minimum log level needed to maintain both system efficiency and effective troubleshooting. • **Configuration principles**: 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. • **Log level**: 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.

Code Completion

Table 5-2 Code completion settings

Item	Description
Code Completion	When enabled, CodeArts Agent provides code completion features, such as real-time suggestions while typing and full code generation via keyboard shortcuts.
Native Completion Conflict Management	When enabled, the IDE's native completion may obscure the suggestions from this product. You are advised to keep this function disabled. **NOTE** This configuration item is displayed only in JetBrains tools (PyCharm, IntelliJ IDEA, WebStorm, and CLion).
Do Not Disturb Mode	When enabled, code completion and chat-related prompts will be hidden to help you focus on coding. You can select specific options based on your needs. In CodeArts Agent IDE and Visual Studio Code, the corresponding configuration items are as follows: **Hide Code Generation Shortcut Hints**/**Hide "Shortcut Key Prompt" Tips**: When enabled, shortcut hints will no longer appear when code is selected. In JetBrains tools (PyCharm, IntelliJ IDEA, WebStorm, and CLion), the corresponding configuration items are as follows: • **Hide "Inline Edit" and "Add to Chat" Prompts**: When enabled, the "Inline Chat" and "Add to Chat" prompts will no longer appear when code is selected. • **Hide Inline Shortcut Hints**: When enabled, inline shortcut hints related to code completion will not be displayed. • **Hide Completion Toolbar**: When enabled, the action toolbar displayed after code completion is generated will be hidden.
Inline Generation	When enabled, CodeArts Agent instantly suggests completions for the current line as you type code (before hitting **Enter**). 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). **NOTE** The tool stops suggesting code if the system faces heavy loads or encounters complex code contexts.

Item	Description
Snippet Generation	When enabled, CodeArts Agent will suggest ways to complete the smallest code block as soon as you hit **Enter**. In JetBrains tools (PyCharm, IntelliJ IDEA, WebStorm, and CLion), you can also adjust the debounce delay to prevent accidental triggers. The debounce delay (ms) sets the waiting time before triggering snippet generation (ranging from 10 to 1,000 ms). **NOTE** The tool stops suggesting code snippets if the system faces heavy loads or encounters complex code contexts.
Intelligent Truncation	When enabled, CodeArts Agent detects the code context and creates full code snippets automatically. Java, C, C++, and Python are currently supported.
Continuation Context Cache	When enabled, cross-file context is automatically cached to avoid repeated extraction and speed up continuation response. **NOTE** This configuration item is displayed only in JetBrains tools (PyCharm, IntelliJ IDEA, WebStorm, and CLion).
Format After Accept	When enabled, the accepted code will be formatted according to the current IDE code style to maintain consistency. **NOTE** This configuration item is displayed only in JetBrains tools (PyCharm, IntelliJ IDEA, WebStorm, and CLion).

Dialog Flow

Table 5-3 Dialog flow configuration items

Category	Item	Description
Chat Settings	Response Language	This setting determines 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: • **Auto**: The model detects the question's language and responds accordingly. • Chinese: The model responds exclusively in Chinese, no matter what language the question uses. • **English**: The model responds exclusively in English, no matter what language the question uses.
	Send Shortcut	**Press Enter to send a message, Ctrl+Enter to start a new line**: After entering a prompt in the input box, press **Enter** to send it. If you want to wrap the line and continue typing, press **Ctrl+Enter**.
		Press Ctrl+Enter to send a message, Enter to start a new line: After entering a prompt in the input box, press **Ctrl+Enter** to send it. If you want to wrap the line and continue typing, press **Enter**.
	Voice Settings	Customize the language displayed after speech-to-text conversion. Select one of the following options from the drop-down list: • Chinese: 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). • **English**: Regardless of whether the voice input is in Chinese, English, or a mix of both, the system automatically converts it into English. **NOTE** This configuration item is displayed only in Visual Studio Code.

Category	Item	Description
Agent	Terminal Command Running Mode	You can 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 the user before execution. • **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 notification.** • **Manual Running**: Before executing any command, the agent sends a confirmation prompt to you. You need to manually confirm it.
	Terminal Command Running Mode > Command Whitelist	This allows you to add prefixes of specific commands to the whitelist as required. Commands added to the whitelist bypass the sandbox mechanism and are executed outside the sandbox. **NOTE** This option is displayed only when the running mode is set to **Running in Sandbox**.
	Terminal Command Running Mode > Custom Sandbox Settings	By configuring the **sandbox.json** file, you can customize files and the network access scopes for processes within the current project's sandbox environment. **NOTE** This configuration item is displayed only when **Terminal Command Running Mode** is set to **Running in Sandbox**.

Category	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. • **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. • **Blocked**: All network connections are blocked, and access to internal or external resources is prohibited. **NOTE** This option is displayed only when the running mode is set to **Running in Sandbox**.
	Auto-approve > Edit	The agent can call tools like edit, write, and deleteFile to edit files on your computer.
	Auto-approve > Open Browser	The agent can access websites in a browser.
	Auto-approve > Web Crawler	The agent can access and capture specified web page content.
Agent Security Policies (This option is displayed only when the running mode is set to **Auto Running** or **Manual Running**.)	Dotfile Protection (Dotfiles)	When enabled, the agent is not allowed to modify configuration files starting with a dot, such as **.env** and **.gitignore**.
	Edit Range 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 Range Protection	When enabled, the agent can only access and read files in the current project directory. It cannot access files in other directories.
Ask	Suggested Questions	When enabled, the system predicts three relevant questions using your preferences and chat history after answering your question. Click these suggestions to ask them instantly.

Agent

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.

For details about built-in agents, see **Built-in Agents**. For details about custom agents, see **Custom Agents**.

Models

In addition to built-in models, enterprise members can also connect third-party large language models to CodeArts Agent. If you have purchased the CodeArts Agent professional edition, you can also add third-party models on the console.

For details, see **Models**.

MCP Tool

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. 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. In addition, multiple out-of-the-box skills are available in the marketplace to quickly boost development efficiency.

- 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: Skills and rules configured based on individual habits or preferences. These settings take effect only for the user who configures them.

For details about skill configuration, see **Skills**. For details about rule configuration, see **Rules**.

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

CodeArts Agent supports repository indexing, helping you quickly search for and obtain related information from the repository. Repository indexes are classified into personal indexes created locally and team indexes created on the cloud.

For details, see **Repository Index**.

Lab Features

Table 5-4 Lab feature settings

Item	Description
Code Optimization	When enabled, the agent automatically identifies issues such as duplication, complexity, performance, and security in the project, and completes structured optimization and repair in one click. **NOTE** This configuration item is displayed only in JetBrains tools (PyCharm, IntelliJ IDEA, WebStorm, and CLion).
Test Case Repair	When enabled, compilation and execution issues in test cases can be fixed. **NOTE** This configuration item is displayed only in JetBrains tools (PyCharm, IntelliJ IDEA, WebStorm, and CLion).
Local LSP	When enabled, the agent can leverage diagnostics, definitions, and symbol data from language services for more precise code comprehension and modification. 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. For details about LSP, 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**.

5.2 Language Settings

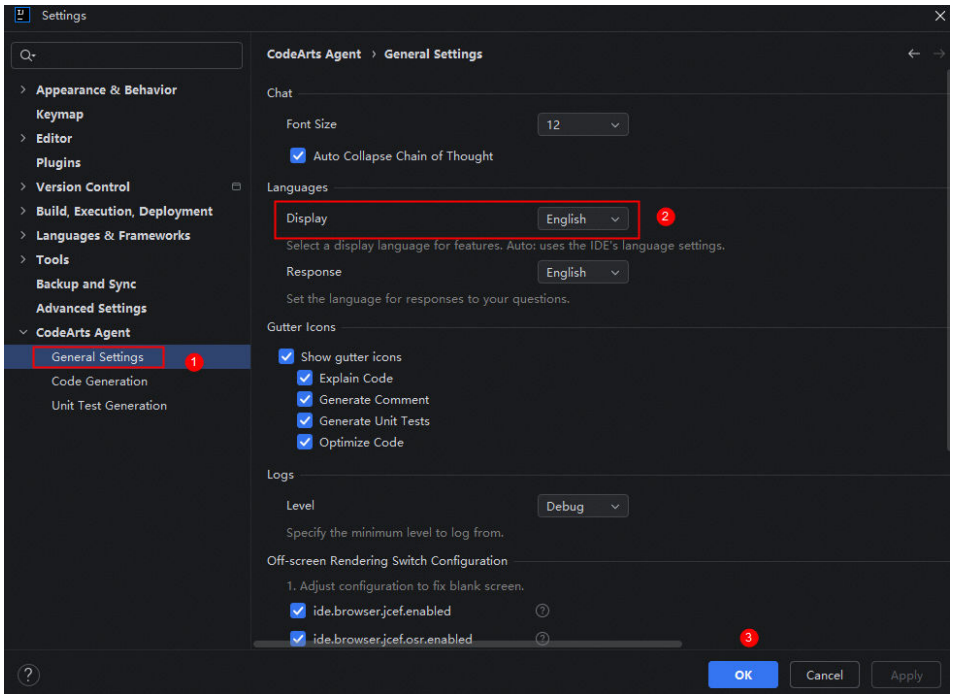
When you install CodeArts Agent for the first time, the extension defaults to English. This section walks you through changing CodeArts Agent's display language.

IntelliJ IDEA

You can change the default language of CodeArts Agent in either of the following ways.

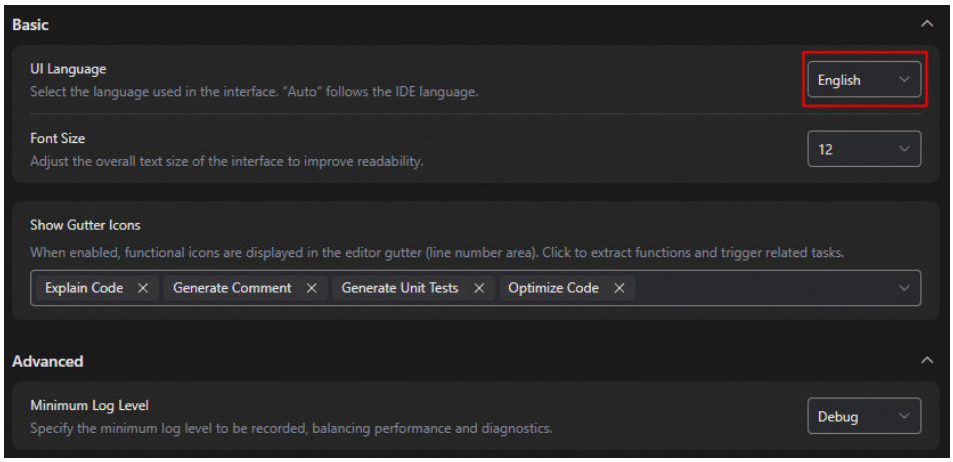
- **Method 1: Reconfiguration in the IntelliJ IDEA Tool Settings**
  a. In IntelliJ IDEA, click  in the upper left corner, and choose **File** > **Settings...**. The IntelliJ IDEA settings page is displayed.
  b. Choose **CodeArts Agent** > **General Settings**. In the **Languages** area, set a desired language.

  Figure 5-1 Languages

  c. Click **OK**. Restart IntelliJ IDEA for the language change to take effect.
- **Method 2: Reconfiguration in the CodeArts Agent Settings**
  a. In the CodeArts Agent chat panel, click  .
  b. Choose **General** > **Basic** > **UI Language** and select a desired language.

Figure 5-2 UI Language



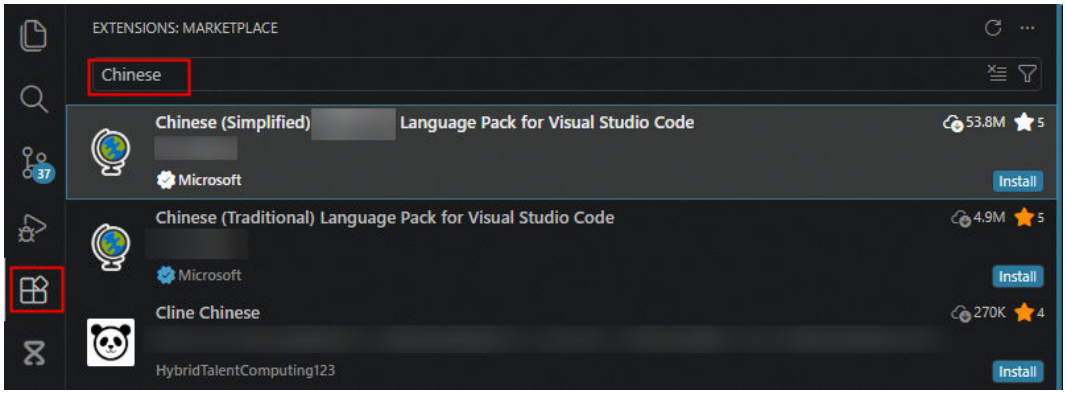
c. Restart IntelliJ IDEA for the language change to take effect.

Visual Studio Code

The extension package for Visual Studio Code uses the same default language as Visual Studio Code itself. Here is how you can switch the language to Chinese.

Step 1 In Visual Studio Code, click  in the left navigation pane, enter **Chinese**, press **Enter**, select the extension package shown in the figure below, and click **Install**.

Figure 5-3 Clicking Install



Step 2 After the extension package is installed, restart Visual Studio Code to complete the language switch.

----End

5.3 Network Proxy

If you are on a corporate network, you may see the error "Model not loaded. Unable to use." If this problem occurs, you can configure a network proxy by referring to this section, restart the tool, and then use the CodeArts Agent extension.

Visual Studio Code

Step 1 In Visual Studio Code, click  in the lower left corner and click **Settings**.

Step 2 Search for **Proxy**, click **Proxy**, and set the **HTTP: No Proxy** and **Http: Proxy** information.

----End

JetBrains

Step 1 In IntelliJ IDEA, click  in the upper left corner, and choose **File** > **Settings...**. The IntelliJ IDEA settings page is displayed.

Step 2 On the **Settings** page, search for **Proxy**, click **HTTP Proxy**, and enter the proxy information as prompted.

CAUTION

If your password contains special characters such as @ and %, escape them before entering the password. Otherwise, the proxy configuration will be invalid.

Step 3 After the proxy configuration is complete, click **Check Connection** and enter any URL to verify the proxy connectivity.

If the message **Connection succeeded** is displayed, the proxy configuration has taken effect.

----End

6 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 an Issue in CodeArts Agent (Recommended)

If you are using Visual Studio Code or JetBrains tools (PyCharm, IntelliJ IDEA, WebStorm, and CLion), you can submit issues in CodeArts Agent directly.

Step 1 Log in to CodeArts Agent as instructed in **Quick Start**.

Step 2 Open the CodeArts Agent chat panel.

- Visual Studio Code: Click  in the Visual Studio Code sidebar to open the CodeArts Agent chat panel.
- JetBrains (with IntelliJ IDEA as an example): Click  in the IntelliJ IDEA sidebar to open the CodeArts Agent chat panel.

Step 3 In the chat panel, click  to open the feedback dialog box.

Step 4 Select a feedback type.

- **Bugs**: This is used to report bugs discovered in the software, including interface errors, function exceptions, and operation freezing.
- **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 5 Enter your feedback by referring to **Table 6-1**.

Table 6-1 Feedback

Parameter	Description
Description	Explain the issue or idea clearly. Keep it no more than 1,000 characters. For AI-related feedback, copy and paste the relevant chat details here.

Parameter	Description
Screenshots	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.
Contact	Provide your contact information to help us better locate issues.
Auxiliary Info	Choose whether to provide plugin information and application logs as required. The information is used only to help locate, reproduce, and resolve issues. • **Plug-in info**: names, versions, and memory usage of all enabled plugins. • **Application logs**: logs generated when CodeArts Agent is used.

Step 6 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.

