

CodeArts TestPlan

User Guide

Issue	01
Date	2026-08-11



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 Working with CodeArts TestPlan.....	1
2 Enabling CodeArts TestPlan.....	4
3 Accessing CodeArts TestPlan Homepage.....	8
4 Configuring a Test Plan.....	9
5 Configuring a Test Version.....	15
6 Configuring a Test Case.....	19
6.1 Generating a Test Case.....	19
6.2 Creating a Test Case by Using Mind Map.....	21
6.2.1 Designing a Test Case on Mind Map.....	21
6.2.2 Creating a Mind Map and Generating a Test Case.....	24
6.2.3 Generating a Combinatorial Test Case on Mind Map.....	34
6.2.4 Managing Mind Maps.....	41
6.3 Creating a Manual Test Case.....	51
6.4 Creating an Automated API Test Case.....	54
6.4.1 Using Automated API Test Cases.....	54
6.4.2 Creating an Automated API Test Case Template.....	55
6.4.3 Adding an API Test Script by Using a Custom URL Request.....	55
6.4.3.1 Adding and Setting the URL Request of an API Script.....	55
6.4.3.2 Setting Checkpoints for an API Script.....	61
6.4.3.3 Setting the Response Extraction of an API Script.....	76
6.4.3.4 Setting Authentication in an API Automation Test Case Script.....	79
6.4.4 Adding an API Script by Importing a Postman File.....	80
6.4.5 Adding an API Script by cURL.....	81
6.4.6 Adding an API Script by Keyword Library.....	81
6.4.6.1 Introduction to Keyword Library.....	81
6.4.6.2 Saving Test Procedure as an API Keyword.....	83
6.4.6.3 Saving Test Procedure as a Combined Keyword.....	87
6.4.6.4 Refreshing a Keyword in Multiple Test Cases.....	89
6.4.7 Adding Logic Control to an API Script.....	92
6.4.8 Setting Test Case Parameters of an API Script.....	95
6.4.9 Setting Environment Parameters of an API Script.....	100

6.4.10 Importing an Automated API Test Case Dataset.....	106
6.4.11 Built-in Functions.....	107
6.4.11.1 Binary Addition Function.....	107
6.4.11.2 Binary Subtraction Function.....	109
6.4.11.3 Binary Multiplication Operation.....	110
6.4.11.4 Binary Division Operation.....	112
6.4.11.5 Obtaining the Current Timestamp.....	115
6.4.11.6 Obtaining a Specified Timestamp.....	116
6.4.11.7 Converting a Date into a Timestamp.....	118
6.4.11.8 Converting a Timestamp into a Date.....	120
6.4.11.9 Timestamp Addition and Subtraction Operations.....	124
6.4.11.10 Generating Base64 Encoding.....	126
6.4.11.11 Generating SHA512 Encoding.....	127
6.4.11.12 Generating SHA256 Encoding.....	129
6.4.11.13 Generating an MD5 Hash Value.....	131
6.4.11.14 Generating a Random Number in a Specified Range.....	132
6.4.11.15 Generating a Number with Specified Decimal Places.....	134
6.4.11.16 Generating a Random String of a Specified Length.....	135
6.4.11.17 Generating a Random Decimal in a Specified Range.....	137
6.4.11.18 Generating a UUID.....	139
6.4.11.19 Custom URL Encoding.....	140
6.4.11.20 URL Decoding.....	142
6.4.11.21 Obtaining an Array via Reverse Index.....	144
6.4.11.22 Obtaining the Element Values of an Array via Reverse Index.....	144
6.4.11.23 Converting Uppercase Letters into Lowercase Letters.....	145
6.4.11.24 Converting Lowercase Letters into Uppercase Letters.....	147
6.4.11.25 Concatenating Strings.....	148
6.4.11.26 Concatenating Strings of Multiple Parameters.....	150
6.4.11.27 Cutting Strings.....	152
6.4.11.28 Obtaining String Length.....	154
6.4.11.29 Extracting a JSON Object.....	155
6.4.11.30 Replacing a String.....	157
6.4.12 System Keywords.....	158
6.4.12.1 Overview.....	158
6.4.12.2 GetIAMToken.....	160
6.4.12.3 MySQLQuery.....	162
6.4.12.4 MySQLUpdate.....	164
6.4.12.5 MySQLInsert.....	166
6.4.12.6 MySQLDelete.....	167
6.4.12.7 OpenGaussQuery.....	169
6.4.12.8 OpenGaussUpdate.....	171
6.4.12.9 OpenGaussInsert.....	172

6.4.12.10 OpenGaussDelete.....	174
6.4.12.11 PostgreSQLQuery.....	175
6.4.12.12 PostgreSQLUpdate.....	177
6.4.12.13 PostgreSQLInsert.....	179
6.4.12.14 PostgreSQLDelete.....	180
6.4.12.15 MongoDBQuery.....	182
6.4.12.16 MongoDBInsert.....	184
6.4.12.17 MongoDBUpdate.....	185
6.4.12.18 MongoDBDelete.....	187
6.4.12.19 RedisGet.....	188
6.4.12.20 RedisSet.....	190
6.4.12.21 OBSWrite.....	192
6.4.12.22 OBSDelete.....	193
6.4.12.23 OBSQuery.....	195
6.4.12.24 KafkaProducer.....	196
6.4.12.25 KafkaConsumer.....	197
6.4.12.26 TCP.....	199
6.4.12.27 UDP.....	201
6.4.12.28 WSConnect.....	202
6.4.12.29 WSRequest.....	203
6.4.12.30 WSWriteOnly.....	204
6.4.12.31 WSReadOnly.....	205
6.4.12.32 WSDisConnect.....	206
6.4.12.33 DubboClient.....	207
6.5 Executing a Test Case.....	211
6.6 Managing Test Cases.....	213
7 Creating and Executing a Test Suite.....	226
7.1 Creating a Test Suite.....	226
7.2 Executing a Test Suite.....	230
8 Viewing and Evaluating Test Quality.....	233
8.1 Viewing the Test Quality Dashboard.....	233
8.2 Creating a Custom Test Report.....	236
8.3 Evaluating Test Quality.....	237
9 Managing CodeArts TestPlan Project Notifications.....	243
9.1 Setting System Messages.....	243
9.2 Setting Email Notifications.....	245
9.3 Setting WeCom Subscriptions.....	247
9.4 Setting DingTalk Subscriptions.....	249
9.5 Setting Feishu Subscriptions.....	251
10 Settings.....	254
10.1 Function Switch.....	254

10.2 Managing Project Members..... 254

10.3 Setting Test Case Fields.....255

10.4 Test Suite Status and Results..... 257

10.5 Background Images and Logos of Test Reports..... 257

10.6 Request Timeout Interval, Resource Pool, and DNS Mapping..... 257

11 Querying Audit Logs..... 259

1 Working with CodeArts TestPlan

CodeArts TestPlan is a one-stop test management platform, covering the entire process of test plan, test design, test cases, test execution, and test evaluation. The platform originates from Huawei's years of high-quality software test engineering methodologies and practices, and aims to help enterprises with collaborative, efficient, and trustworthy test activities before product release.

CodeArts TestPlan is an individual service provided within the [CodeArts](#) solution. For details about its role in the solution, see [CodeArts service portfolio](#).

Basic Operation Process

Figure 1-1 Basic operation process

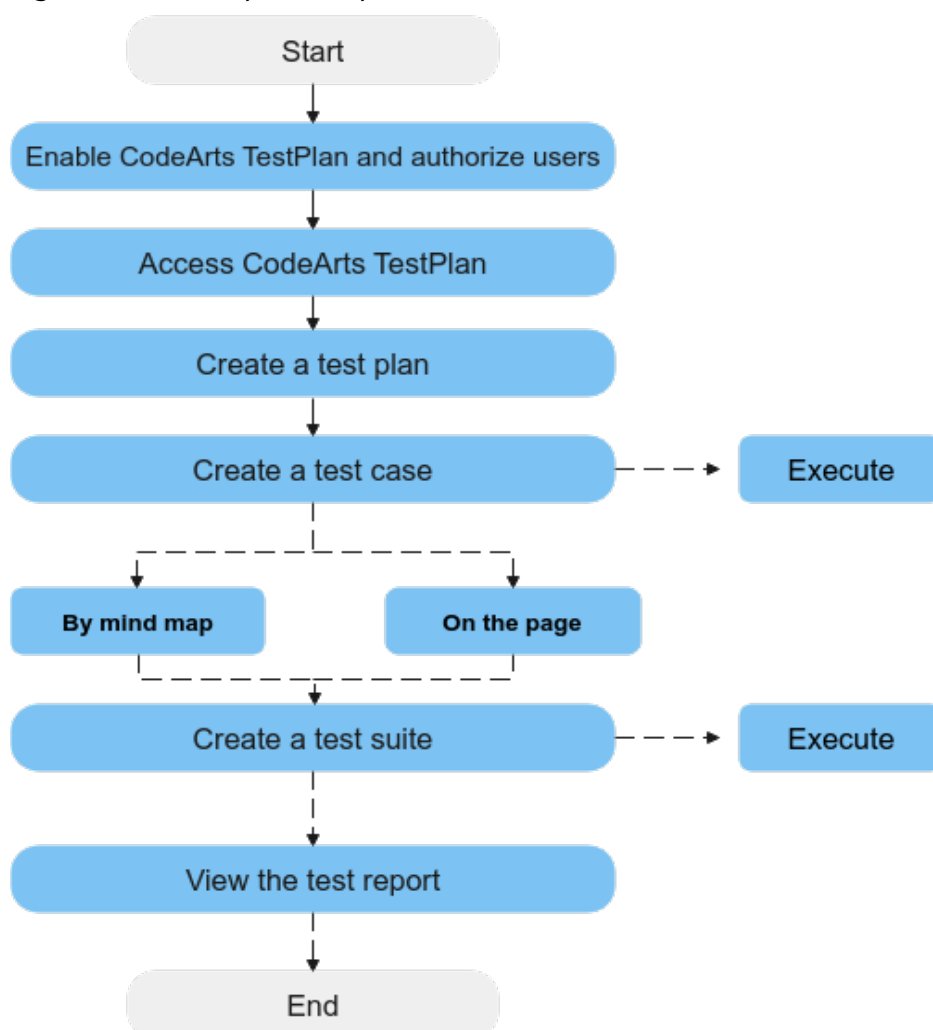


Table 1-1 Description of the operation process

Step	Description
Enable and authorize CodeArts TestPlan	Purchase CodeArts TestPlan and configure permissions for CodeArts TestPlan project-level roles .
Access CodeArts TestPlan	Access the CodeArts TestPlan homepage .
Create a test plan	Configure a test plan .
Create a test case	Create test cases by mind map or on the Testing Case page , and execute and manage test cases.
Create a test suite	Create and execute test suites.

Step	Description
View the test quality report	View the test report and assess test quality .

2 Enabling CodeArts TestPlan

Prerequisites

You have registered with Huawei Cloud and completed real-name authentication. If you do not have a HUAWEI account yet, follow these steps to create one:

1. Visit [Huawei Cloud official website](#).
2. Click **Sign Up** and create your account as instructed.
Once your account is created, the system automatically redirects you to your personal information page.
3. Complete individual or enterprise real-name authentication. For details, see [Real-Name Authentication](#).

Purchasing CodeArts TestPlan

Refer to [Purchasing CodeArts](#).

Adding Members and Assigning Roles

Path for project-level permissions:

Step 1 Click a project name to enter the project. If no project exists, [create one](#).

Step 2 In the navigation pane, choose **Settings > Permissions**.

----End

The following table describes the user roles and their operation permissions on CodeArts TestPlan.

Table 2-1 User roles and permissions

Operation /Role	Project Administrator	Project Manager	Test Manager	Product Manager	System Engineer	Committee	Developer	Tester	O&M Manager	Participant/Viewer
Test cases	Create, modify, delete, execute, stop, and view test cases. Import test cases from files or versions. Export test cases. Check specification compliance. Merge test cases to the baseline (except product manager).						Create, modify, delete, execute, stop, and view test cases. Import test cases from files or versions. Export test cases. Check specification compliance. Cannot merge test cases to the baseline.		Export and view test cases.	
Test suites	Create, modify, delete, view, execute, and stop test suites.								View test suites.	
Test versions	View test versions. Create, modify, and delete test versions (except product manager).						View test versions.			
Test plans	Create, modify, delete, and view test plans.								View test plans.	
Quality reports	Create, modify, delete, and view quality reports.								View quality reports.	
Quality assessment	Create, modify, delete, view, and download quality assessments.								View and download quality assessments .	
Test case recycle bin	Delete, restore, and view items in the recycle bin.								View items in the recycle bin.	

Operation/Role	Project Administrator	Project Manager	Test Manager	Product Manager	System Engineer	Committee	Developer	Tester	O&M Manager	Participant/Viewer
Settings	View test settings. Create, modify, and delete settings (except product manager, developer, and tester).								View test settings.	
API test keywords	Create, view, delete, and edit keywords.								View keywords.	
Global variables	Create, view, delete, and edit global variables.								View global variables.	
Mind maps	View and create mind maps, and delete and modify personal mind maps. Delete and modify all mind maps (except product manager, developer, and tester).								View mind maps.	
Mind map backups	View and create mind maps, and delete and restore personal mind map backups. Delete and restore all mind map backups (except product manager, developer, and tester).								View mind map backups.	
Mind map templates	View and create mind map templates, and delete and modify personal mind map templates. Delete and modify all mind map templates (except product manager, developer, and tester).								View mind map templates.	
Test design recycle bin	View deleted mind maps, and delete and restore personal mind maps in the recycle bin.								View deleted mind maps, and delete personal mind maps in the recycle bin.	
Mind map Asset	View factor libraries. Create, modify, import, export, and delete factor libraries (except product managers, developers, and testers).								View factor libraries.	
Mind map Review	Create, view, modify, and delete mind map reviews. Delete all mind map reviews (except developers and testers).								View mind map reviews.	

 NOTE

To modify the permissions of the current role, contact the project administrator.
Custom roles have no preset permissions. Contact the project administrator to add operation permissions for custom roles.

3 Accessing CodeArts TestPlan Homepage

Step 1 [Log in to the Huawei Cloud console.](#)

Step 2 Click  in the upper left corner of the page and choose **Developer Services > CodeArts TestPlan** from the service list.

Step 3 Click **Go to CodeArts TestPlan**.

Step 4 In the upper left corner of the page, click the project drop-down list, select the target project, and click **Enter Project**.

Alternatively, click **Homepage** on the top navigation bar, hover the cursor over the target project card, and click **Testing**. The **Testing Plan** page is displayed.

----End

4 Configuring a Test Plan

In the development phase, CodeArts TestPlan helps test teams formulate comprehensive and detailed test plans based on project requirements and product features.

Test tasks are split and assigned to testers with deadlines. As the testing goes as planned, team members can track the progress of tasks, such as the number of test cases completed and being executed. Test results, including the quantity, type, and severity of bugs, are recorded for timely analysis so that testers can make adjustments and keep the testing process smooth and efficient. This provides a solid guarantee for product quality.

The typical testing process involves creating a test plan, designing tests, executing them, and generating a test report.

- In the test planning and design stages, define the test scope and objectives, and develop strategies, tools, environments, models, test cases, and automated test scripts.
- A test plan specifies the test time, scope, and objectives, and manages all test activities. It can be specific to a version, sprint, or project.

Prerequisites

You have obtained the permissions to create and modify test plans in the target project. For details, see [Table 2-1](#).

Notes and Constraints

1. For Scrum and IPD projects, work items that can be added as requirements to test plans include epics, features, and stories. For Kanban projects, the default requirement work items can be added to test plans as requirements.
2. A single test plan can be associated with a maximum of 300 requirements.

Creating a Test Plan

Step 1 [Access the CodeArts TestPlan homepage](#) through a project.

Step 2 In the navigation pane, choose **Testing > Testing Plan**.

Step 3 Click **Create Plan** in the upper part of the page. The **Create Test Plan** page is displayed.

Step 4 Configure the following information and click **Next**.

Configuration Item	Description
Name	Name of a test plan. Describe the test scenario or function. Enter 3 to 128 characters. Use only letters, digits, spaces, and special characters (,-_/*&`^~;(){}=+,x...!@#\$.%.\'[]<>?"—). Do not start with a space.
Version (optional)	The version of the test plan. Enter 3 to 32 characters of digits, letters, hyphens (-), underscores (_), and periods (.).
Processor	Select a member in the project as the processor of the test plan.
Plan Period	Select the start date and end date of the test plan.
Release (optional)	This parameter is available when the project type of the plan is IPD-Standalone Software or IPD-System Device .
Sprint (optional)	Associate the test plan with a sprint of the project.
Description (optional)	Brief description of a plan. The value contains a maximum of 1,000 bytes.

Step 5 Select the required execution mode and requirements (max. 300 requirements), and click **Save**.

- The execution mode selected here can be modified in the test plan later.
- The **Testing Case** and **Testing Execution** pages will show the tabs for the selected modes. Click the tabs and then handle test cases and suites for manual or automated API testing. The **Quality Report** page will also show relevant statistics.

Step 6 Check the test plan in the test plan list in card mode.

----End

Copying a Test Plan

Step 1 Hover over the new test plan card and click .

Step 2 In the displayed dialog box, update the name, processor, and execution period for the new test plan as required.

Step 3 After completing the configuration, click **OK** to copy the test plan.

----End

Editing a Test Plan

In the test plan list, click the name of the test plan to be edited. The editing window is displayed on the right of the page.

- On the **Details** tab page, edit the test plan (including the test plan name, description, execution mode, and basic information) and click **Save**.
- On the **Requirements** tab page, add or delete the requirements of the current test plan.
- On the **Test Cases** tab page, view the test cases in the plan, and add test cases from the version that the test plan belongs to.
- On the **Operation History** tab page, view the editing history of the test plan.

Designing a Test Plan

Design test cases, develop automated testing scripts, and prepare test data based on specified test requirements.

Step 1 Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Plan**.

Step 3 Select a test plan to be designed from the list.

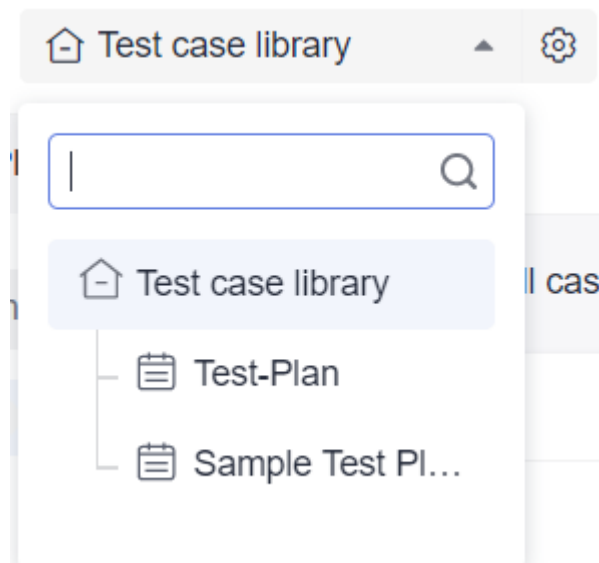
- Hover over **Design**. The design progress is displayed, including mind map count, covered requirements, total requirements, total test cases, and requirement coverage.
- Click **Design** to go to the **Testing Case** page.

Step 4 Click a required test type tab and then click **Create Case** to create a test case.

After a test case is created, the **Design** circle changes from gray to blue, indicating that the test plan is under design. For details about how to create a test case, see [related topics](#).

Step 5 Click the test plan name in the upper left corner of the page to switch to another test plan, or view the global test case library.

The global test case library displays all test cases of all test plans in the current version. You can maintain the global test case library as required.



-----End

Helpful Links

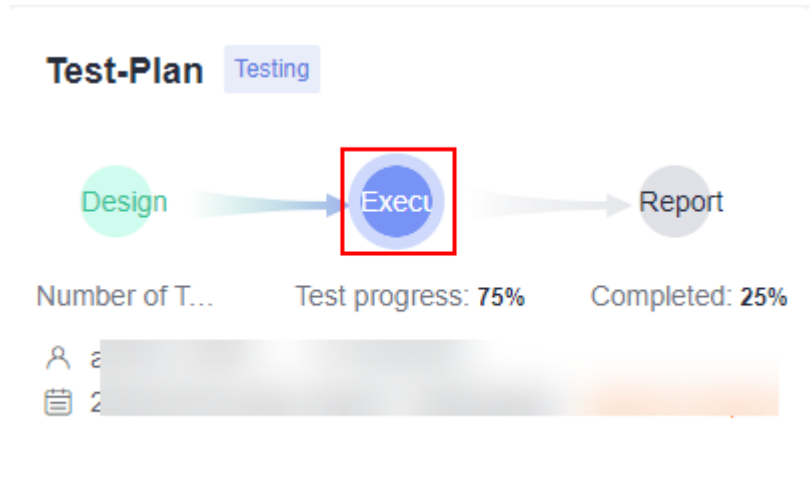
- [Creating a Manual Test Case](#): Create manual test cases on the **Manual Test** tab page.
- [Creating an Automated API Test Case](#): Create automated API test cases on the **Auto API Test** tab page.

Executing a Test Plan

- Step 1** Return to the **Testing Plan** page and select the test plan to be executed from the list.
- Move the cursor over **Execute** to view the execution progress of the test plan. Statistics include the test progress, number of executed test cases, pass rate, and completed bugs/total bugs.
 - Click **Execute** to go to the **Testing Execution** page.

- Step 2** Create a test suite to execute test cases in batches.

After a test case is executed, the **Execute** circle changes from gray to blue, indicating that the test plan is under test.



For details about test execution operations, see [related topics](#).

Step 3 Set the status of all test cases in the test plan to **Completed**. Then the status of the test plan is automatically updated to **Completed**.

----End

Helpful Links

- [Creating a Manual Test Suite](#): Execute multiple test cases in a test suite on the **Manual Test** tab page.
- [Executing an Automated API Test Suite](#): Execute the automated API test suite on the **Auto API Test** tab page.

Managing and Measuring a Test Plan

Step 1 Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Plan**.

Step 3 Move the cursor over the test plan whose report you want to view, and click **Report**. The test quality report page of the current project is displayed. View the report or create a new. For details, see [Evaluating Test Quality](#).

Step 4 Move the cursor over the test plan card and click **View Quality Dashboard** in the upper right corner.

- The page displays the current requirement coverage, defects, case pass rate, and case completion rate of the test plan. You can analyze and record test risks on the page.
- In the **Manual Test** and **Auto API Test** areas, statistics on the execution of test cases and the number of bugs are displayed by execution type.
- To add more reports, click a report type in the lower left corner and click **Add Report**, or click **Create Report** in the upper right corner.
- To switch to another test plan or view the quality report of the global test case library, click the test plan name in the upper left corner of the page and select the target plan.

----End

Helpful Links

Viewing the Test Quality Dashboard: Check the test quality dashboard. It shows the statistics on requirements and test cases in the current plan.

5 Configuring a Test Version

In CodeArts TestPlan, test cases are managed and evaluated by product baseline, branch version, and test plan. The service enables teams to collaborate efficiently, allows for the reuse of test assets across different versions, and supports the simultaneous development and delivery of multiple product versions.

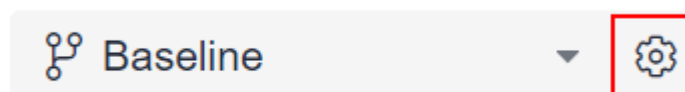
- **Version:** A version records the test data of a software version, containing details on managing and running test cases and suites, as well as quality reports.
- **Relationship between version and test plan:** A version contains multiple test plans (and one test case library).
- **Baseline:** A baseline is a special version that has been formally reviewed and is the basis for subsequent test activities. Cases in a baseline are generally stable and are test assets accumulated for a long time.
- **Hierarchical management of test cases:** Test cases can be managed by layer, that is, by product baseline library, version branch, and test plan. You can merge test cases from a version to the baseline, import test cases from the baseline to a version, and handle conflicts when merging cases between versions. This supports collaborative and simultaneous testing of multiple versions, and enables accumulation and reuse of test assets.

Creating a Version

Step 1 Log in to the service homepage, search for your target project, and click the project name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Case**.

Step 3 Click  on the right of **Baseline**. The **Version Management** page is displayed.



Step 4 Click **Additions**.

Step 5 Enter a version name and click **Save**.

----End

Editing a Version

On the **Version Management** page, move the cursor over the test version to be edited, click , and change the version name.

Deleting a Version

On the **Version Management** page, click  in the **Operation** column of a version to delete it.

Viewing Test Case Changes

On the **Version Management** page, click  in the **Operation** column of a version, and view the change history.

Managing Versions

Based on your test strategy, import the test cases from the baseline to a version, or merge the test cases in a version to the baseline.

- **Importing Test Cases from Other Versions**

Step 1 Log in to the service homepage, search for your target project, and click the project name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Case**.

Step 3 Select a version from the version drop-down list in the upper left corner of the page.

Step 4 Click the **Manual Test** tab. In the right area of the tab page, choose **Import > Import from Version**.

 NOTE

If no test case is available or you need to create a test case, click **Create Case**. For details, see [Creating a Manual Test Case](#).

Step 5 Select **Test Cases** for **Resource Type** to import test cases from another version together with the directory of them. If you deselect **Test Cases**, only the directory will be imported.

Step 6 Select a source version from the drop-down list under **Source**.

Step 7 In the displayed dialog box, select the test cases to be imported, select an overwriting rule, and click **OK**.

To select test cases in batches, move the cursor over the check box on the left of **Name**, and select **Select More**. In the displayed dialog box, specify a range of test cases to be imported or select all test cases.

----End

Merging Test Cases in a Version into the Baseline (On Testing Case Page)

- Step 1** Log in to the service homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** Select a version from the version drop-down list in the upper left corner of the page.
- Step 4** Select a test type tab. In the right area of the tab page, click **Merge into Baseline**.

NOTE

If no test case is available or you need to create a test case, click **Create Case**.

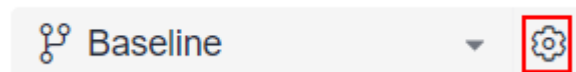
- Step 5** In the displayed dialog box, select the test cases to be merged, select an overwriting rule, and click **OK**.

To select test cases in batches, hover over the check box on the left of **Name**, and select **Select Current Page** or **Select All Pages** to select the test cases in the current page or all pages.

----End

Merging All Test Cases in a Version into the Baseline (Through Version Management)

- Step 1** Log in to the service homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** Click  on the right of **Baseline**. The **Version Management** page is displayed.



- Step 4** Select the version whose cases you want to merge into the baseline, and click  in the **Operation** column.
- Step 5** In the displayed dialog box, select a rule and click **OK** to merge test cases.

----End

Measuring a Version

- Step 1** Log in to the service homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Quality Report**. The **Quality Report** page is displayed.
- Step 3** Check the quality report of the version.

To view the quality reports of other versions or test plans, click the drop-down list boxes in the upper left corner of the page and select the desired version or test plan.

----End

6 Configuring a Test Case

6.1 Generating a Test Case

A test case describes a test task for a specific software product, including the test solution, method, technique, and policy. The content includes the test objective, test environment, input data, test procedure, expected result, and test script.

In CodeArts TestPlan, you can create test cases for manual testing, automated API testing, and custom automated testing.

You can directly create test cases, or design the test on a mind map and then generate test cases on the map.

Concepts related to test case:

- **Test case library:** It stores all types of cases in different versions of a project. For details, see [Test Case Library](#).
- **Manual test case:** It includes test steps for a specific test scenario, their expected results, the test case result, and the test case status.
- **Automated API test case:** It is used to automate the testing of predefined APIs to verify system functions and performance. These test cases simulate user operations and input data based on API design standards to check whether the system processes requests correctly and returns accurate responses.

Test Case Library

In the test case library, you can view, manage, and use all test cases of the versions of a project.

- You can add test cases from the test case library to test plans.
- You can view, manage, and use only the test cases in the current test plan.
- The test cases created in test plans are collected to the test case library.

To view the test case library, perform the following steps:

- Step 1** On the top navigation bar, choose **Testing > Testing Case**. The test case library is displayed by default.

Select a test plan from the drop-down list of the test case library to view the test case details of the plan.

----End

Generating Test Cases Based on Test Features

You can create a feature folder in the **Test Features** directory and then create test cases in the folder.

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Click **Test Features** on the left of the page.

- By default, all test cases belong to this **Test Features** directory, which has a preset default subdirectory **Other**. (Note that the **Other** folder cannot contain additional subdirectories). Click  on the right of the **Test Features** directory to create another subdirectory. Click  next to the new subdirectory to delete or rename the subdirectory, or create test cases or subdirectories.

Step 3 Click the subdirectory, select an execution type tab, and click **Create Case**. Then, the created test case is saved in the subdirectory.

----End

Filtering Test Feature Directories

Click  on the right of the test feature directory and select one of the following criteria to display the directories you want.

- Hide empty directories: Only the directories that contain test cases are displayed.
- Show empty directories only:
 - If a subdirectory is empty and its parent directory contains cases, both of them are displayed, and their numbers of cases are displayed.
 - If the parent directory and subdirectory are empty, both of them are displayed, and their numbers of cases are displayed.
 - If the parent directory is empty and the subdirectory contains cases, neither of them is displayed.
 - If both the parent and subdirectories contain cases, neither of them is displayed.
- Sort A to Z: Display directories by name in ascending or descending order.

Filtering Test Cases by Test Feature Directory

Above the test case list, enable or disable the toggle switch for collapsing cases of subdirectories.

Moving a Test Feature Directory

Right-click the test feature directory you want to move and choose **Move to Folder**. In the displayed dialog box, select the destination directory and click **OK**.

Deleting a Test Feature Directory

Click  on the right of a directory to delete it. After the directory is deleted, the test cases in it will be moved to the **Other** directory.

When you delete an empty test feature directory from a version's test case library, a confirmation dialog will appear for the first time. If you check **Do not show again**, this confirmation dialog will not be displayed for future deletions.

Generating Test Cases Based on Test Requirements

If a project has been associated with a requirement, you can create test cases for the requirement in the **Test Requirements** directory.

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Click **Test Requirements** on the left of the page. By default, all associated requirements of the project belong to this **Test Requirements** directory.

To associate a project with requirements, [create a work item](#).

Step 3 Click a requirement in the **Test Requirements** directory, all test cases associated with the requirement are displayed. To create a test case, click an execution type tab and click **Create Case**. The new test case is associated with the requirement by default.

Step 4 On the right of the requirement name, click  and choose to view the requirement details or create a test case for the requirement.

----End

6.2 Creating a Test Case by Using Mind Map

6.2.1 Designing a Test Case on Mind Map

CodeArts TestPlan provides a test design feature that offers multi-dimensional heuristic test strategies and design templates. Testers can break down test requirements into scenarios, test points, and test cases. In this way, we hope to inspire testers to design tests creatively, visualize their ideas, and improve test coverage. This feature enables continuous optimization of test completeness, and helps testers reduce product test omissions during execution.

The purpose of test design is to clarify the scope, objectives, and methods of test activities, guide test activities, and standardize test behaviors.

With the help of mind map, you can perform heuristic and visual test design.

NOTE

A 65-day trial is available now. After the trial ends, create a service ticket or contact the account manager to apply for the feature.

Notes and Constraints

- On test design mind maps, you can edit nodes, expand all nodes with one click, and export mind maps to PDF and PNG files. The display effect depends on how well your browsers render the web pages, and is also restricted by the

browser version, kernel, memory, style, node content, and mind map size. In extreme cases, the nodes may expand slowly, lines may be lost, the export operation may be suspended or canceled, or the export effect may be poor. If these happen, split the mind map into multiple smaller ones or use other browsers. You are advised to use less than 1,000 nodes in a mind map. If the browser still freezes when there is only a small number of nodes, wait until the export is complete. If you use the MeSign browser, test design mind maps cannot be copied due to its kernel and function restrictions, and the export quality of PDF and PNG files is not optimal. You are advised to use Google Chrome or the 360 browsers instead.

- If a red indicator appears in the upper-right corner, your changes may not be saved yet due to slow loading or background processing. Wait until the green indicator appears before continuing to edit the mind map.

Designing a Mind Map Based on a Requirement

Create a mind map for a requirement, break down test scenarios, analyze test points, and output test plans and test cases based on the requirement.

Step 1 Search for your target project and click the name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Design**.

Step 3 Click **Test Requirements** on the left of the page. By default, all associated requirements of the project belong to this **Requirements** directory.

To associate a project with requirements, [create a work item](#).

Step 4 In the search box under **Test Requirements**, enter a keyword of the target requirement and click . The target requirement is displayed in the requirement directory.

Step 5 Click **Test Requirements** on the left, select a requirement, and click **Create** or **Templates** in the upper left corner.

Step 6 The mind map design page is displayed. For subsequent operations, go to [Creating a Mind Map and Generating a Test Case](#).

----End

Designing a Mind Map Based on a Feature

Create a mind map for a feature, break down test scenarios, analyze test points, and output test plans and test cases based on the feature.

Step 1 Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Design**.

Step 3 Click **Test Features**. In the search box under it, enter a keyword of the target feature and click .

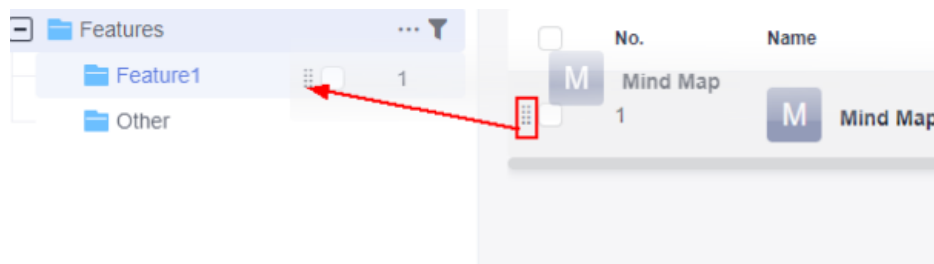
Step 4 Select a test feature and click **Create** or **Templates**.

Step 5 The mind map design page is displayed. For subsequent operations, go to [Creating a Mind Map and Generating a Test Case](#).

----End

Moving a Mind Map to Change Its Test Feature

Step 1 In the test design list, move the cursor to the check box of a mind map.  is displayed on the left of the check box. Drag the mind map to the required feature in the **Test Features** directory on the left. The feature that the mind map is designed for is changed.



Step 2 Check the mind map in the directory of the new feature.

To move mind maps in batches, select multiple or all mind maps, click **Batch Change Feature**. In the displayed dialog box, select the target directory, and click **Confirm**.

NOTE

Mind maps cannot be moved to other requirements in the **Test Requirements** directory.

----End

Associating a Test Case Generated on a Mind Map with Requirements

After a test case is designed and archived for a test feature on a mind map, associate the case with the test requirements.

Step 1 Go to the mind map designed based on a test feature.

Step 2 Find a test case that has been archived and needs to be associated with a requirement, and click .

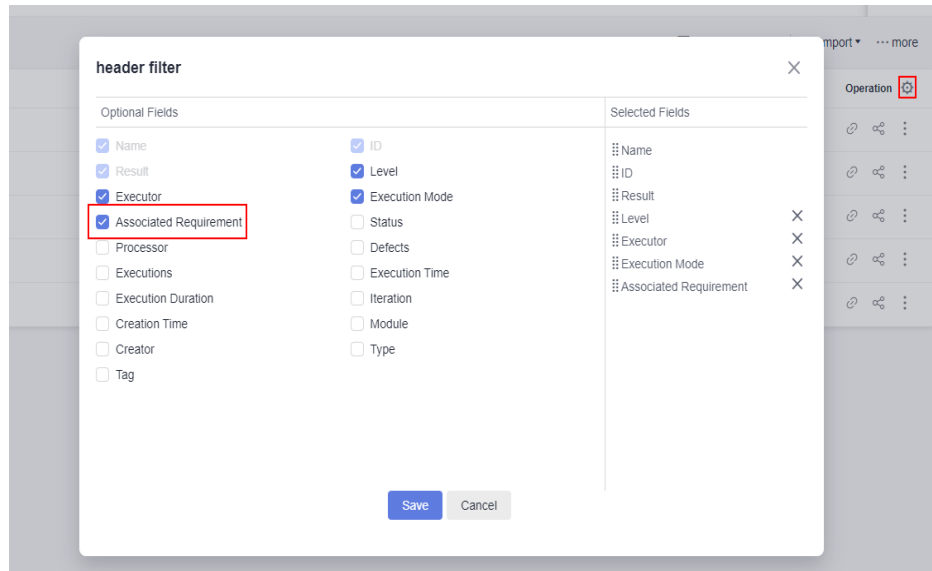
Step 3 Click the **Requirements** tab and click **Associated With Requirement**.

Step 4 In the displayed dialog box, select the target requirement in the current plan and project, and click **OK**.

Step 5 Click **Save** in the upper right corner.

Step 6 In the navigation pane, choose **Testing > Testing Case**. On the displayed page, find the test case. The value in the **Associated Requirement** column is **Associated**.

If the **Associated Requirement** column does not exist in the list, click the gear icon in the upper right corner, select **Associated Requirement**, and click **Save**.



----End

6.2.2 Creating a Mind Map and Generating a Test Case

Mind maps, also called brain maps, are used to plan test schemes, design test scenarios, define test points, orchestrate test steps, and generate test cases. You can use the mind map function on the test design page.

Prerequisites

You have been assigned a role (except the viewer, participant, and O&M manager) that has permissions to create mind maps in the target project. For details, see [Adding Members and Assigning Roles](#).

Notes and Constraints

The description of a single node in a mind map can contain a maximum of 5,000 characters.

Creating a Mind Map from Scratch

Step 1 Search for your target project and click the name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Design**.

Step 3 Click **Create** in the upper left corner of the page.

The new mind map page is displayed. The root node named **Mind Map** by default is displayed in the middle of the page. You can double-click the root node to change its name.

Step 4 In the upper left corner of the page, click , rename the mind map, and click **Confirm**.

Step 5 Click  in the upper left corner of the page. The test design list is displayed. The created mind map is displayed in the list.

----End

Creating a Mind Map from Template

Step 1 Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Design**.

Step 3 Click  next to **Create** and choose **Templates**.

Step 4 Select a required template. Click **Preview** to view the mind map details. Click **Use** to access the mind map.

Step 5 View the displayed mind map page and details about the selected template.

Step 6 Click  in the upper left corner of the page. The test design list is displayed.

Step 7 To open multiple mind maps on multiple tab pages, click  in the **Operation** column and choose **Open in New Tab**.

----End

Renaming a Mind Map

Step 1 In the mind map list, click  in the **Operation** column.

Step 2 The name of the mind map is displayed in the dialog box. Enter a new name. The length cannot exceed 500 characters.

Step 3 Click **OK**. The mind map is renamed.

----End

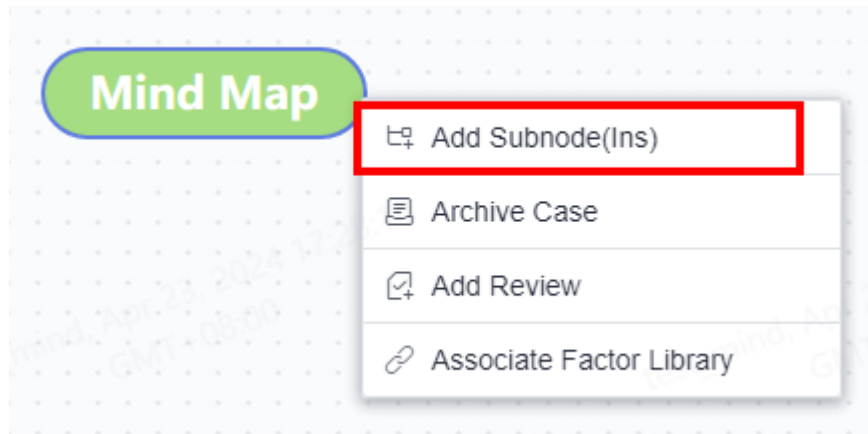
Drawing a Mind Map

On the **Testing Design** page, click the name of a mind map to be edited, and draw the mind map.

Adding a Node

Add new sibling nodes and subnodes. Select any node in the mind map and select the type of the new node as required. Only subnodes can be added to the root node. Sibling nodes and subnodes can be added to other nodes.

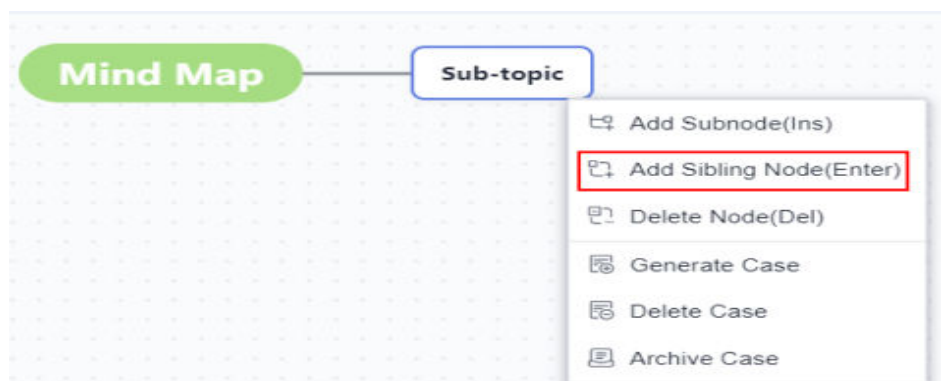
- Add a subnode.
Access a mind map, select a node, and add a subnode in either of the following ways:
 - Right-click and choose **Add Subnode (Ins)** from the shortcut menu.
 - Press **Insert** or **Tab**.



- Add a sibling node.

Access a mind map, select any node except the root node, and add a sibling node in either of the following ways:

- Right-click and choose **Add Sibling Node (Enter)** from the shortcut menu.
- Press **Enter**.



Deleting a Node

You can delete any node except the root node in the mind map. If the deleted node contains subnodes, the subnodes are also deleted.

Access a mind map, select any node except the root node, and delete the node in one of the following ways:

- Right-click and choose **Delete Node (Del)** from the shortcut menu.
- Click  in the upper right corner of the page and select **Delete Current Node**.
- Press **Delete**.

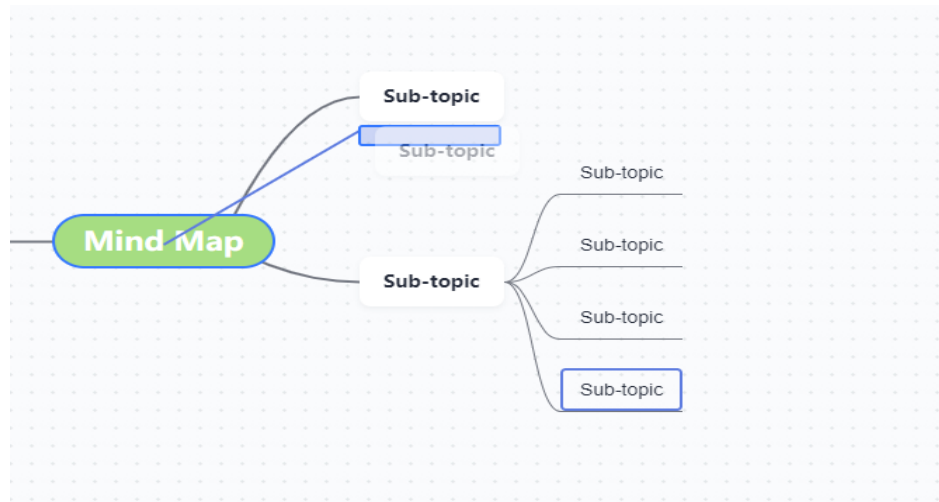
Moving a Node

- Move a node up or down.

Access a mind map, select the node to be moved, and click  or  on the toolbar above the mind map to move the node up or down among its sibling nodes.

- Drag a node.

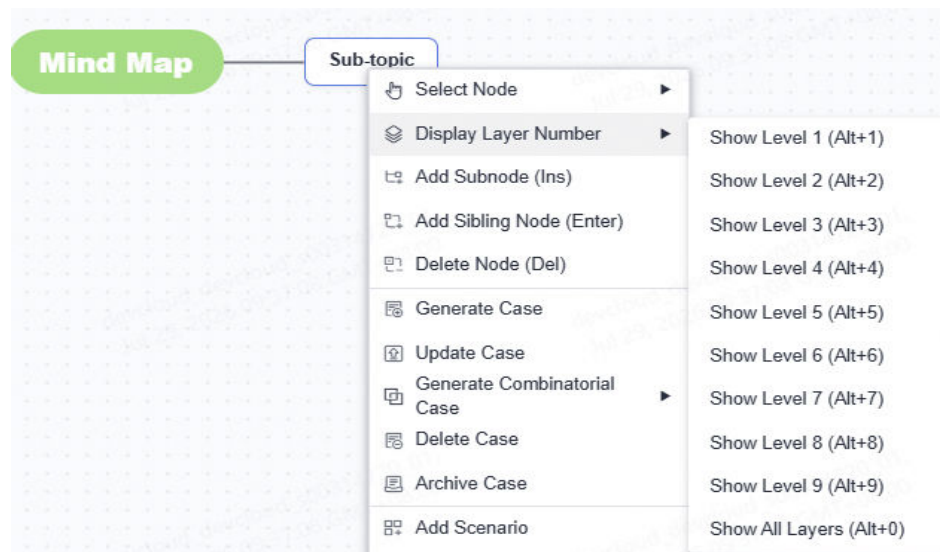
Access a mind map, select the node to be moved, hold down the left mouse button, drag the node to the required position, and release the mouse.



Expanding Display Levels

Expand to the required mind map node level.

Right-click a node, choose **Display Layer Number** from the shortcut menu, and choose the target level.



Designing a Scenario

Design scenarios based on the requirement for which the mind map is created.

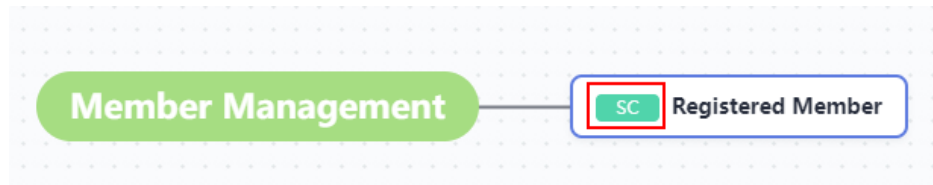
Step 1 Go to a mind map.

Step 2 Add a subnode to the root node. (You can set any non-root node as a scenario.)

Step 3 Enter a scenario description. For example, **Membership registration**.

Step 4 Select the node created in **Step 2** and click  on the toolbar above the mind map.

If **SC** is displayed on the selected node, the scenario is added successfully.



----End

Designing a Test Point

Design test points based on the requirement for which the mind map is created.

Step 1 Go to a mind map.

Step 2 Add a subnode to the root node. (You can set any non-root node as a test point.)

Step 3 Enter a subnode name. For example, **Membership registration**.

The test point name can contain 1 to 128 characters. Only letters, digits, and special characters (-_/*&`'^~;:(){}=+,x...—!@#\$.[]<>?-"') are supported.

Step 4 Select the node created in **Step 2** and click **TP** on the toolbar above the mind map.

If **TP** is displayed on the selected node, the test point is added successfully.



----End

Designing a Test Case

Design a test case to refine a test point.

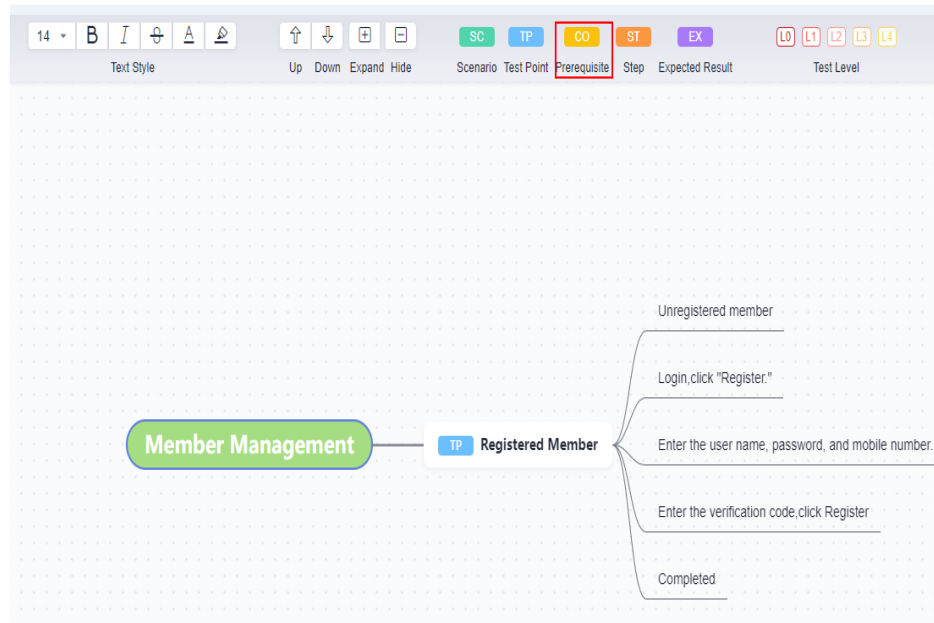
A test case consists of prerequisites, steps, and expected results. Set the three parts as subnodes of the test point node.

Step 1 Go to a mind map.

Step 2 Under the **Membership registration** node, create subnodes and enter the descriptions of prerequisites, steps, and expected results.

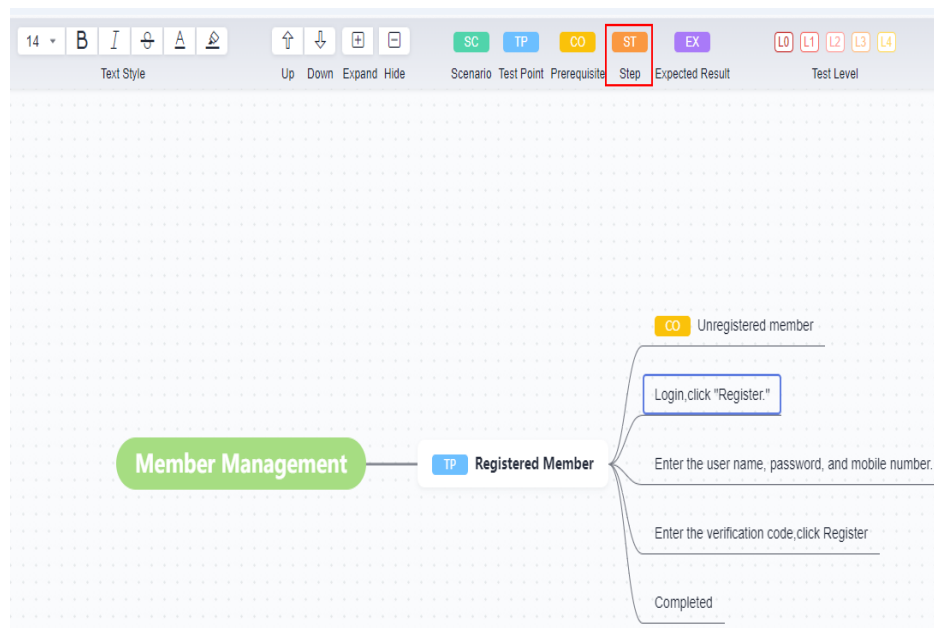
Step 3 Select a node to which you have entered a prerequisite, and click **CO** on the toolbar above the mind map.

If **CO** is displayed on the node, it is set as a prerequisite node.



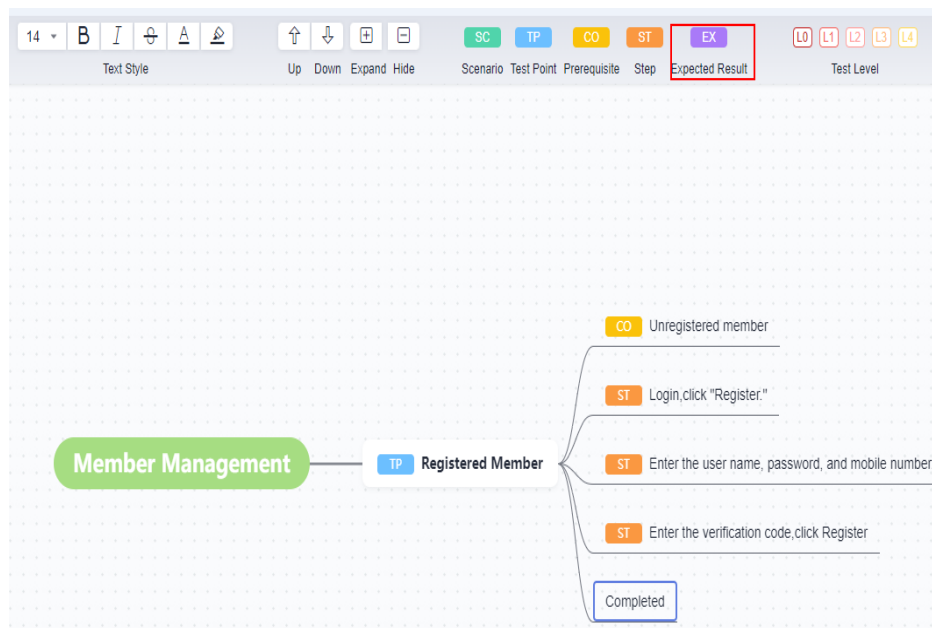
Step 4 Select a node to which you have entered a step, and click **ST** on the toolbar above the mind map.

If **ST** is displayed on the node, it is set as a step node.



Step 5 Select a node to which you have entered an expected result, and click **EX** on the toolbar above the mind map.

If **EX** is displayed on the node, it is set as an expected result node.

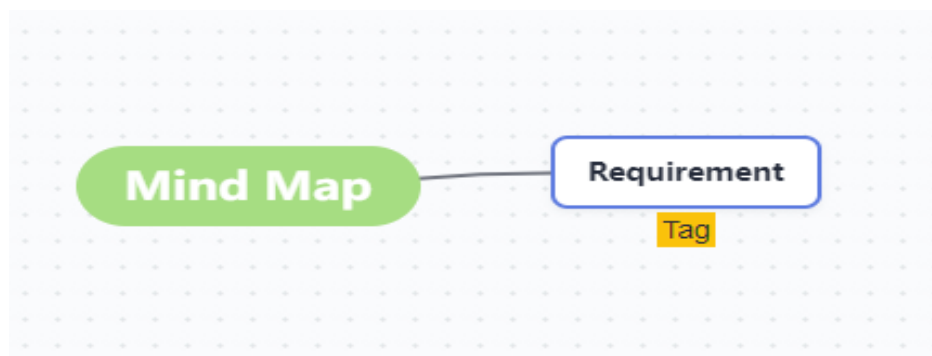


----End

Adding a Tag

Add tags to nodes as auxiliary information. Only one tag can be added to a node. The tag length must be 1 to 128 characters.

- Step 1** Access a mind map, right-click a node, and choose **Add Tag**.
- Step 2** Enter the tag content in the text box under the node and click any position on the screen to save.
- Step 3** After the settings are saved, a tag with a yellow background is displayed under the node.



----End

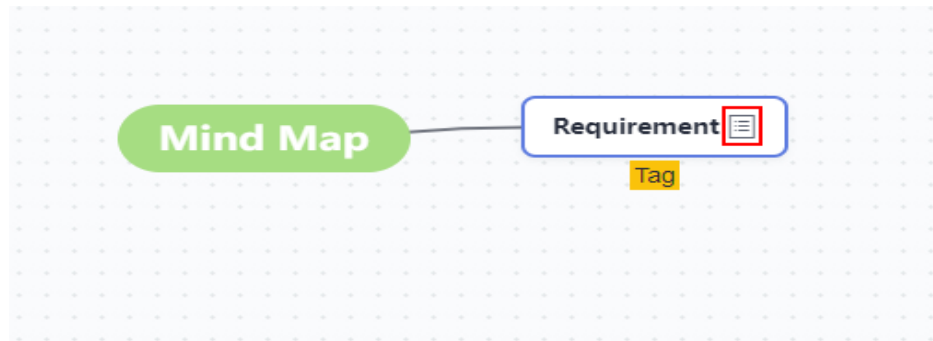
Adding Remarks

Add remarks to nodes as auxiliary information. Only one remark can be added to a node. The remark length must be 1 to 500 characters.

- Step 1** Access a mind map, select a node, and click **Remarks** in the upper part of the page.

Step 2 Enter remarks in the window that is displayed on the right of the page and click anywhere on the screen to save.

Step 3 If  is displayed on the node, the remarks are successfully added to the node.



----End

Performing Online Review

Review and comment on mind map nodes.

Step 1 Access the mind map to be reviewed, right-click a test case node, and choose **Add Review**.

Step 2 In the displayed dialog box, select a type and enter comments.

Step 3 Select reviewers. A maximum of five reviewers can be selected.

Step 4 Click **Add Review**.

Step 5 View the review record in the historical review comments.

Step 6 On the right of the review record, click . Then edit the content in the **Type** and **Comments** boxes. (Only the creator and reviewers of the review can edit and delete it.)

To close a confirmed review, toggle off the switch on its right.

----End

Generating Test Cases

After test case design, generate test cases on the mind map.

Generating a Single Test Case

Step 1 In the mind map, right-click the scenario (SC) or test point (TP) node for which a test case is to be generated, and choose **Generate Case**.

Step 2 Check whether  is displayed on the node. This icon indicates that a draft test case is generated.

Step 3 Click . The case details drawer page is displayed.

Before generating cases, enable or disable the function of generating each step with an expected result. For details, see [Function Switch](#).

 NOTE

1. When a test case is generated from a TP node, only the first-layer step (ST) subnodes are read, from top to bottom.
2. There will be no expected result (EX) displayed if not set for an ST node.
3. If an ST node has multiple EX nodes, only the first one will be generated as the expected result.

Step 4 To generate a new test case after you modify the previous nodes, repeat the preceding steps.

----End

Generating Cases in Batches

If a scenario has multiple test points, generate draft test cases in batches for it.

Step 1 In the mind map, select a scenario node that contains multiple test points.

Step 2 Right-click the node and choose **Generate Case** from the shortcut menu.

The  icon is displayed on all test point nodes of the scenario node, indicating that draft test cases have been generated for them.

----End

Updating a Case

To update draft test cases after you modify the mind map, perform the following operations.

Step 1 Modify the mind map and right-click the target SC or TP node.

Step 2 Click **Update Case**.

Step 3 In the displayed dialog box, select the draft test cases to be updated.

Step 4 Click **Confirm**.

----End

Configuring the Test Case ID Rule

Step 1 Click  in the upper right corner of the page.

Step 2 In the displayed dialog box, enter the prefix, auto-increment start value, and connector (optional).

Step 3 Click **OK**.

To modify the rule, click  in the dialog box.

Step 4 [Generate test cases](#). Test case IDs meeting the rule will be displayed in the test case details page.

 NOTE

1. To modify an ID, double-click it in the test case details page.
2. The ID will start from the value you set and increase each time a test case is generated.
3. The existing IDs will not be affected by the rule configuration. The new rule applies only to test cases generated after the configuration.

----End

Archiving Cases

Archive draft test cases. Archived test cases will be displayed on the **Testing Case** page.

Step 1 In the mind map, right-click a node for which test cases have been generated and choose **Archive Case** from the shortcut menu. The **Archive Case** window is displayed.

Step 2 On the left, select the test cases to be archived. On the right, set the version, test case library/test plan, and execution mode.

If you select the SC node for archiving, the **Synchronize directories** toggle will be available. Turning it on syncs the SC-to-TP node path with the directory on the **Testing Case** page. This ensures test cases created from the TP node are saved there. If you leave it off, only the TP node's test cases are archived and stored in the test feature directory you pick later.

 NOTE

The SC node directory cannot be synced to the **Other** directory in the test feature directory. If you select **Other**, only the test cases generated from the TP node are archived regardless of whether **Synchronize directories** is enabled.

Step 3 Associate the requirement with a test feature directory.

Step 4 Click **OK**. The archiving progress bar is displayed.

Step 5 Click the  icon in the node to find the target test case.

Step 6 Click . The test case details page is displayed.

----End

Deleting Archived Test Cases from a Mind Map

Step 1 Right-click the node with test points for which test cases have been generated.

Step 2 Choose **Delete Case** from the short-cut menu.

Step 3 Select one of the following options:

- **Do not delete them.:** Only the test case formats will be removed from the mind map. The test cases remain in the list in the **Testing Case** page.
- **Label them as "Discarded".:** In the manual test case list, the cases will be tagged with **Discarded** in the basic information.

- **Delete them.:** Those in the case library and test plan will also be deleted. The associations with test suites, requirements, and bugs cannot be restored.

Step 4 Enter **DELETE** in the text box and click **OK**.

----End

6.2.3 Generating a Combinatorial Test Case on Mind Map

You can combine test factors to generate test cases. Data factor combinations enable full coverage of test factors for generation of test cases, and diversify test design functions.

Test factors refer to the factors that affect a test, such as the environment, test mode, and test difficulty. The number of test factors of a test is the number of factors that affect the test. By combining these factors, each test case can cover multiple conditions, which helps avoid missing tests.

Prerequisites

You have created a **Test Point** node, for example, **Test the basic mobile phone functions**. Note that combination can be done only on test points.



By Adding Action and Data Factors

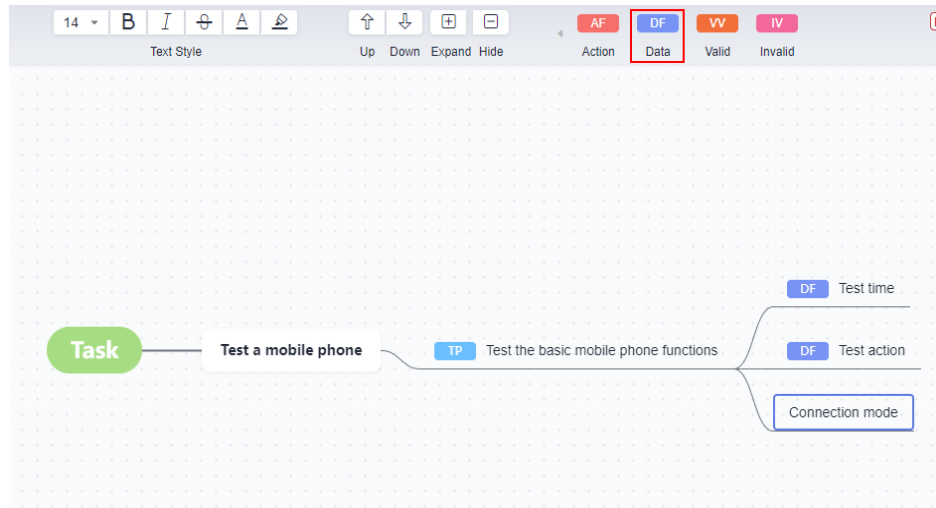
During test design, you can add **Action Factor**, **Data Factor**, **Valid Value**, and **Invalid Value** to mind map nodes.

Step 1 Go to a mind map.

Step 2 On the top toolbar bar, click  on the right of **Expected Result**. The **Action Factor**, **Data Factor**, **Valid Value**, and **Invalid Value** tag icons are displayed.

Step 3 Select the node to which you want to add action or data factors (max. 100 action factors and 100 data factors per mind map).

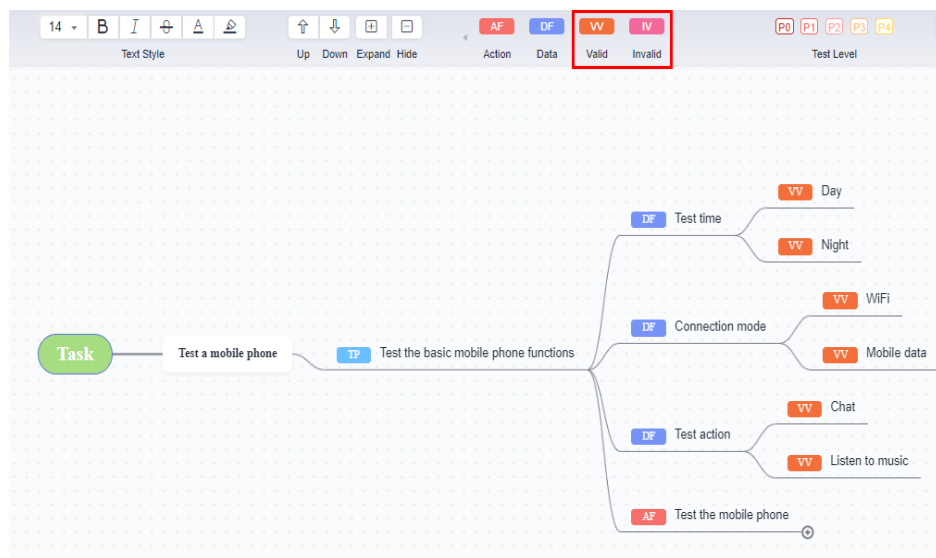
Click  on the toolbar above the mind map. Take the node **Test the basic mobile phone functions** as an example. If the test process is to use a mobile phone to perform *{test action}* in *{connection mode}* at *{test time}*, the data factors can be *{test time}*, *{connection mode}*, and *{test action}*, to cover all scenarios.



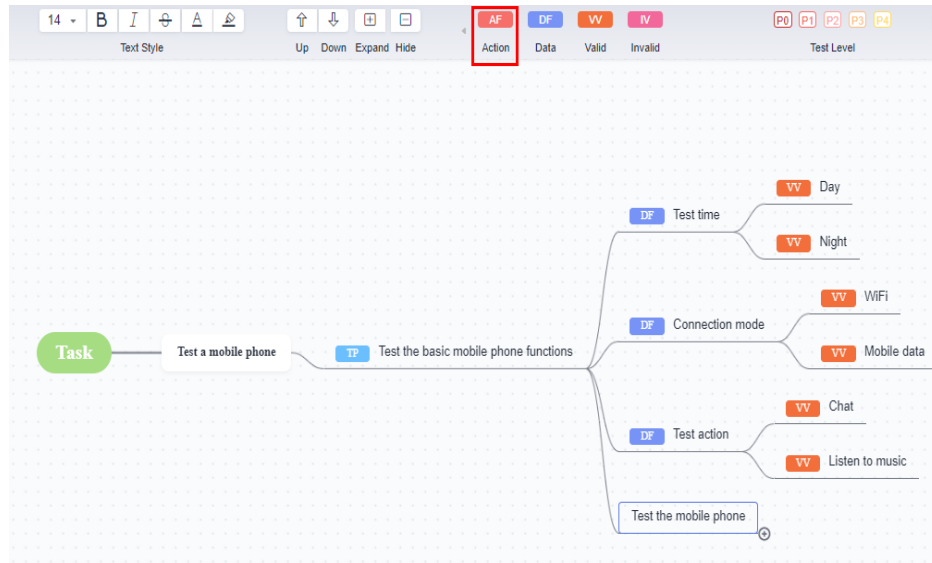
Step 4 Add a valid or invalid value: Add subnodes to the data factor nodes, select the subnodes, and click the valid value icon **VV** or the invalid value icon **IV** on the toolbar.

- **Valid Value:** A normal system value of a data factor to be tested.
- **Invalid Value:** An abnormal system value to be tested, or a value not within the range of a data factor. A good test model should include both valid and invalid values to test whether the system program can correctly process errors. Only one invalid value is allowed in a test case. There is no need for multiple invalid values as any single one will trigger system exceptions.

The following figure shows the valid and invalid value subnodes of the data factor nodes **Test time**, **Connection mode**, and **Test action**.



Step 5 Add an action factor: Select a subnode and click the action factor icon **AF** on the toolbar. For example, set **Test the mobile phone** to be an action factor.



Step 6 Right-click a test point node and choose **Generate Combinatorial Case > Generate** from the shortcut menu.

NOTE

The system finds the parent node of the DF nodes and displays their valid and invalid values from top to bottom.

Step 7 On the **Generate Combinatorial Case** page, select the factors you need. By default, all action and data factors are selected.

Step 8 Click **Next**.

Step 9 Hover over a combinatorial algorithm and click **Use and Preview**. For details about the combinatorial algorithms, see the following table.

Data Combination Coverage Type	Description
All Combinations (AC)	All values of each test factor are combined. AC is the most comprehensive coverage mode.
N-Wise	This combination covers N inputs. When N equals the number of parameters, this mode generates AC. Tests prove that when $N=2$ (pair-wise), the generated test data provides the highest efficiency, making this one of the most widely used algorithms.
PairWise	Pair-wise testing is N-wise testing when N is 2.
TripleWise	Triple-wise testing is N-wise testing when N is 3. It generates more combinations than pair-wise testing.
Basic Choice (BC)	Use this algorithm if you want to compare single-factor changes. You need to set a base combination first. From the base combination, new combinations are created by varying one factor value at a time.

Data Combination Coverage Type	Description
Each Choice (EC)	Each test factor value appears at least once in any combination.

To change the algorithm, click **Change Algorithm** and select another algorithm.

If you return to the mind map and modify or delete a test factor node, click **Generate Combinatorial Case** and then click **Refresh** to apply your changes.

Step 10 (Optional) Add data factor constraints to limit the values to be used for generating combinatorial cases.

1. Select **Data Constraint** and click **Add**. You can add up to 20 constraints.
2. Select the required operation relationship, set **Variable** to the data factor to which the constraint is to be added, select the operator, and set **Value** to the valid or invalid value of the data factor.

To clear the value, click .

3. In the condition expression box, check the configured algorithm and click **Preview Constraint** to view the constraint.
4. Click **Confirm**.
5. If you modify the data constraints, click **Refresh** in the upper right corner of the combination preview.

Example variables and values:

Type: Primary, Logical, Single, Span, Stripe, Mirror, RAID-5

Size: 10, 100, 500, 1000, 5000, 10000, 40000

Format method: quick, slow

File system1: FAT, FAT32, NTFS

File system2: FAT, FAT32, NTFS

Cluster size: 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536

Compression: ON, OFF

- **Non-conditional constraints**

To limit parameter values, use non-conditional constraints.

[Size] > 10000: The value of **Size** in the case set can only be **40000**.

[File system1] <> [File system2]: The values of **File system1** and **File system2** in the case set must be different.

[Size] > 10000 and [Compression]="ON": The value of **Size** in the case set can only be **40000** and the value of **Compression** can only be **ON**.

- **IF condition constraints**

To set an IF condition, use this format: **IF** *expression1* **THEN** *expression2*, where the expressions are in the format *[Data factor] Operator [Value]*.

For example, the expression **IF [File system1]="FAT" THEN [Cluster Size] <= 4096** indicates that if **File system1** is **FAT**, then the value of **Cluster Size** can only be **512, 1024, 2048, or 4096**.

- **AND/OR/NOT**

For complex expressions, use AND, OR, and NOT, and then enclose them with parentheses (required).

For example, the expression **IF ([File system1]="NTPS" OR ([File system2]="NTPS" AND [Cluster Size] <= 4096)) THEN [Compression]="OFF"** indicates that if either **File system1** or **File system2** is **NTPS**, and **Cluster Size** is **512, 1024, 2048, or 4096**, then **Compression** is **OFF**.

- **IN/LIKE**

To limit a factor value within a specific value set, use the IN operator.

For example, the expression **IF [File system1] IN {"FAT","FAT32"} THEN [Cluster Size] <= 4096** indicates that if **File system1** is **FAT** or **FAT32**, the value of **Cluster Size** can only be **512, 1024, 2048, or 4096**. This constraint is equivalent to **IF ([File system1]="FAT" OR [File system1]="FAT32") THEN [Cluster Size] <= 4096**.

For wildcard matching, use the LIKE operator together with asterisk (*) or question mark (?). For example, use **FA***, ***FA**, and **?A** to match the values starting with FA, ending with FA, and containing A, respectively.

For example, the expression **IF ([File system1] LIKE "FA*") THEN [Cluster Size] <= 4096** indicates that if **File system1** is set to **FAT** or **FAT32**, the value of **Cluster Size** can only be **512, 1024, 2048, or 4096**.

NOTE

1. Constraints differentiate between integer and string variables. If all values of a variable are numbers, it is regarded as a number. If any one of the values is not a number, the variable is regarded as a string.
2. When an operator is used to constrain two data factors in the format *[Data factor 1] Operator [Data factor 2]*, both data factors must be of the same type.
3. Do not use the LIKE operator to constrain two integer variables. Do not use operators **>**, **<**, **>=**, and **<=** to constrain two string variables.

Step 11 In the **Combination Preview** list, select the data combination to be created.

Move the cursor over the first check box, select **Select All Pages** or **Select Current Page**, and click **Next**.

Step 12 On the **Configure Test Case** page, set the following parameters:

- **Test Case:** The default case name is in the format of *TP node name_\${Data factor}*.
- **ID:** The case IDs will be generated based on the auto-increment rule you set.
- **Case Level:** Select **P0**, **P1**, **P2**, **P3**, or **P4**.
- **Test Case Description:** Enter a description. Max. 500 characters.
- **Prerequisite:** Enter the prerequisites. Max. 2,000 characters.

Step 13 Write test steps. Use **\$** to reference data factors. Click **Confirm**.

Test Steps

#	Step Description	Expected Result	Operation
⋮ 1	Use the phone to perform \${Test action} in \${Connection mode} at \${Test time}.		+

Step 14 Click  to view the test case list.

Step 15 The generated test cases are displayed by default. Choose **Combination Strategies** to view the history of combinatorial algorithms.

Step 16 Click  in the **Operation** column of the test case to be archived, and click **Archive Cases**. Other operations:

- Archive multiple test cases: Select multiple test cases and click **Archive Cases**.
- Edit a test case: Click  in the **Operation** column, and view and edit the test case.
- Delete a test case: Click  in the **Operation** column and click **Delete**.
- Delete test cases in batches: Select multiple test cases and click **Delete**.
- Search for a test case: Enter a keyword in the search box, and click .
- Filter test cases: Click the drop-down list box under **Test Cases**, and select **All**, **Unarchived**, or **Archived**.

Step 17 On the **Archive Case** page, select the target test cases on the left. In the right area, select the version, test plan, execution mode, and test feature directory, and click **Confirm**.

Step 18 In the navigation pane, choose **Testing > Testing Case**. Select the required version, test plan, and test type tab, and view the archived test cases.

-----End

By Factor Library

Creating a Factor

Step 1 In the navigation pane, choose **Testing > Testing Design**.

Step 2 Click **Test Factor Center** in the upper right corner of the page.

Step 3 Create a directory if there is only the root directory: Click  next to the root directory and click **Create Directory**.

Step 4 Enter a subdirectory name within 1 to 500 characters.

Step 5 Click the directory and click **Create Factor**.

Step 6 Configure the following information and click **Confirm**.

Configurati on Item	Mandat ory	Description
Factor Name	Yes	Max. 500 characters.
Factor Type	Yes	Data Factor or Action Factor .
Factor Description	No	Brief description of a factor. Max. 500 characters.
Prerequisites	Yes	Prerequisites for action factors.
Test Steps	Yes	Step descriptions and expected results for action factors. Max. 5,000 characters. Click + in the Operation column to add more test steps.
Data Type	No	The default value is String .
Valid Value/ Invalid Value	No	Valid or invalid values for data factors. Click Add to add more valid or invalid values.
Remarks	No	Max. 500 characters.

Step 7 The new factor is displayed in the directory. You can:

- Copy a factor: Click  in the **Operation** column. On the displayed page, modify the factor information and click **Copy**.
- Edit a factor: Click  in the **Operation** column. On the displayed page, modify the factor information.
- Delete a factor: Click  in the **Operation** column and click **Delete**.
- Delete factors in batches: Select factors and click **Delete** on the tool bar below.
- Export factors: Select factors, click **Export** on the tool bar below, and save the file to the local PC.
- Filter factors: Click the drop-down list box on the right of **Create Factor** and select **All Factor**, **My Factor**, **Action Factor**, or **Data Factor**.
- Search for a factor: Type a keyword in the search box and click the search icon.
- Edit the factor list headers: Click  in the **Operation** column of the factor list. In the displayed dialog box, select headers and adjust the display sequence on the right.
- Import: Click **Import** in the upper right corner. In the displayed dialog box, click **Download Template** to download the template to the local PC. Edit the template file, click  in the dialog box, select the file, and click **OK**.

----End

Associating a Mind Map with the Factor Library

Step 1 In the navigation pane, choose **Testing > Testing Design**.

Step 2 Create or select a mind map.

Step 3 Right-click a node and choose **Generate Combinatorial Case > Associate Test Factor** from the shortcut menu.

 NOTE

Association can be done only on test point nodes.

Step 4 On the **Associate Factor Library** page, select factors and click **Confirm**. The factors are displayed in the mind map.

Step 5 For details about how to generate a case, see [Step 6](#).

----End

6.2.4 Managing Mind Maps

Notes and Constraints

A maximum of 20 MB mind map files can be imported. A maximum of 100 mind maps in a compressed package can be imported.

Basic Operations on Mind Map

In addition to editing nodes, you can perform the following operations.

Operation	Description
Rename	Click  in the upper left corner of the mind map. In the displayed dialog box, rename the mind map.
Delete a mind map	In the upper right corner of the mind map page, click  , select Delete Mind Map , and click Confirm to delete the current mind map. The deleted mind map is moved to the recycle bin.

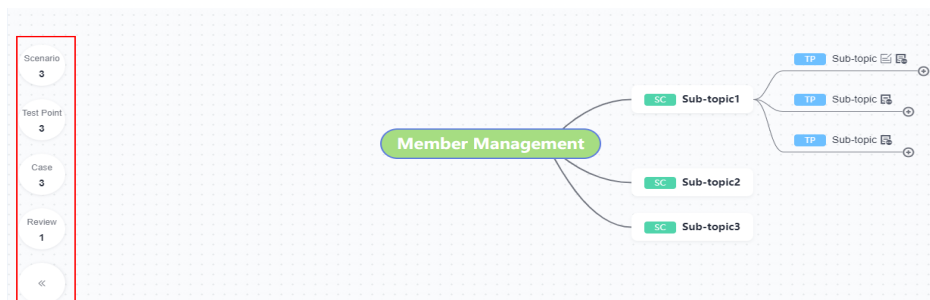
Operation	Description
Manage the recycle bin (on mind map)	Click  in the upper right corner of the mind map and select Recycle Bin from the drop-down list. In the displayed dialog box, view the list of deleted mind maps. For mind maps in the recycle bin, you can perform the following operations: - Click  and Confirm to restore the mind map in the corresponding row. NOTE - Restoring the mind map will replace the content on the current page. Exercise caution when performing this operation. - Once a record in the recycle bin is restored, the record is removed from the list. - You are advised to create a blank mind map to restore the mind map in the recycle bin. - Click  to view details about the mind map in the corresponding row. - Click  and Confirm to permanently delete the mind map in the corresponding row. Deleted mind maps cannot be restored. Exercise caution before performing this operation.
Recycle bin (on the Testing Design page)	In the lower left corner of the Testing Design page, click Recycle Bin. In the displayed dialog box, see the list of deleted mind maps. For mind maps in the recycle bin, you can perform the following operations: - Click  and Confirm to restore the mind map in the corresponding row. NOTE - The mind map will be restored to the directory. - Once a record in the recycle bin is restored, the record is removed from the list. - Click  to view details about the mind map in the corresponding row. - Click  and Confirm to permanently delete the mind map in the corresponding row. Deleted mind maps cannot be restored. Exercise caution before performing this operation.

Operation	Description
Import	Click  on the toolbar above the mind map, and then click Confirm. In the displayed dialog box, select a local .xmind file to import. After the import, the following items will be displayed on the page: priority icon, flag icon, uploaded image, uploaded file, text style, node tag, and node remarks. The following node name prefixes will be converted into node type icons on the page: SC: (scenario), TP: (test point), CO: (prerequisite), ST: (step), EX: (expected result), TP#P*: (case level). NOTE • A file whose size does not exceed 20 MB can be imported. • A maximum of 100 .xmind files in a compressed file can be imported. • The imported content will replace the mind map on the current page. Exercise caution when performing this operation. • After the mind map is imported successfully, the root node is not changed.
Export	Click  on the toolbar above the mind map, select the PDF, PNG, HTSM, or Xmind format, and click Confirm to export the mind map to the local computer. • For PDF or PNG, expand all nodes before exporting to capture the complete mind map. • For HTSM, only mind maps of 20 MB or less can be exported. • For Xmind, the exported .xmind file (max. 20 MB) can be opened using Xmind 2020 or later. These items cannot be exported: the  icon (indicates that test cases have been generated), test case information, case level tags on non-TP nodes, and node review content. The icons of these node types are converted into text prefixes: scenario (SC:), test point (TP:), case level (TP#P*:), prerequisite (CO:), step (ST:), expected result (EX:), action factor (AF:), data factor (DF:), valid value (VV:), and invalid value (IV:). You can format selected text within a mind map node so that a single node contains multiple text styles. However, the Xmind format supports only one text style per node. If a node contains two or more text styles, none of its text styles will be exported.
Backup	Click  on the toolbar above the mind map. In the displayed dialog box, enter the name and description, and click Confirm to create a backup for the mind map on the page.

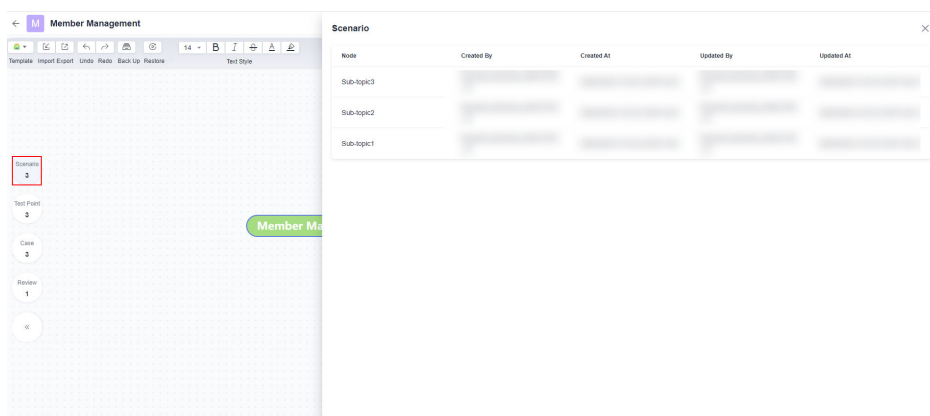
Operation	Description
Restore	Click  on the toolbar of the mind map. In the displayed dialog box, select a backup and click . In the displayed dialog box, click Confirm to restore the backup mind map to the current page. NOTE - Restoring the mind map will replace the content on the current page. Exercise caution when performing this operation. - Before the mind map is restored, the content of the current mind map is automatically backed up.
Find/Replace	Click *** > Find/Replace in the upper right corner of the page, or press Ctrl+F on the page. In the displayed dialog box, enter the search term and its replacement, click Replace to change the node names one by one, or click Replace All to change them all at once. To find and replace among the nodes you selected, click Selected nodes.
Drag-select nodes	Right-click and drag to select multiple nodes. You can then perform batch operations on them, for example, changing the node type to test point, scenario, or expected result, or removing the type.
Undo	Click  on the toolbar above the mind map to cancel the last operation on the page.
Redo	Click  on the toolbar above the mind map to restore the latest undone operation.
Expand all	Select a node with  and click  on the toolbar above the mind map to expand all subnodes under the node. The icon next to the node changes to .
Collapse all	Select a node with  and click  on the toolbar above the mind map to collapse all subnodes under the node. The icon next to the node changes to .

Collecting Node Statistics

On the left of the mind map page, the statistics of each type of node are displayed, including the total number of scenarios, test points, cases, and reviews.



Click any statistical value. A drawer expands from the right, showing the node details.



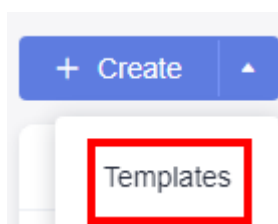
Click any node name in the list to locate the corresponding node in the mind map and view the node details.

Managing a Mind Map Template

Saving a Mind Map as Template

A mind map can be saved as a template.

- Step 1** Access a mind map and edit the mind map as required.
- Step 2** Click **Template** on the toolbar of the mind map and choose **Save As**.
- Step 3** In displayed the dialog box, enter a name and click **Confirm**.
- Step 4** Click  in the upper left corner to return to the test design list. Click **Templates** in the upper left corner of the page, and click the **Custom Templates** tab. The saved template is displayed in the dialog box.



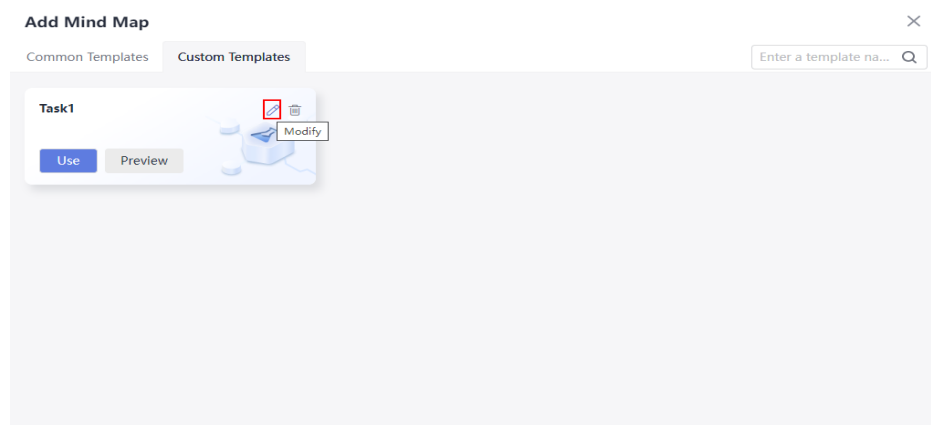
----End

Editing a Template

Members can edit the templates saved by themselves.

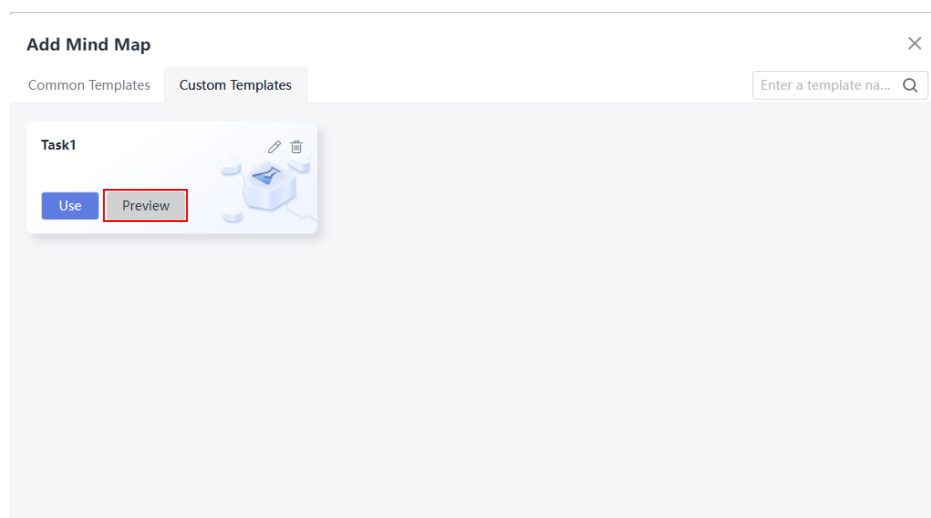
Step 1 Go to the **Testing Design** page and click **Templates** in the upper left corner of the page.

Step 2 Click the **Custom Templates** tab. In the displayed dialog box, select a template and click .



Step 3 Edit the template as required. After the editing is complete, click  in the upper left corner of the page.

Step 4 Click **Templates** in the upper left corner of the page. Click the **Custom Templates** tab. In the displayed dialog box, locate the edited template and click **Preview** to view the modified template details.

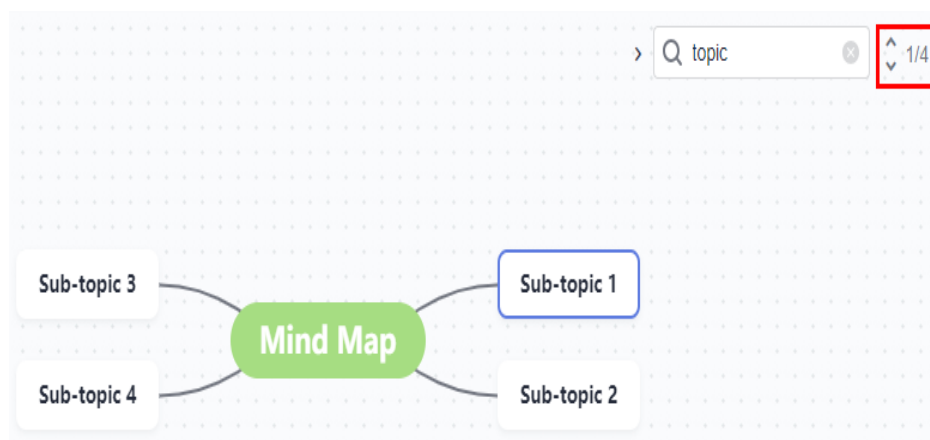


----End

Searching for Nodes

In the upper right corner of a mind map, enter a keyword in the search box and click  to find the node containing the keyword. The found node is marked in a blue box.

On the right of the search box, you can see the number of matched items and the sequence number of the matched item of the current node. You can click the up or down arrow to shift to other matching items.



Renaming a Node

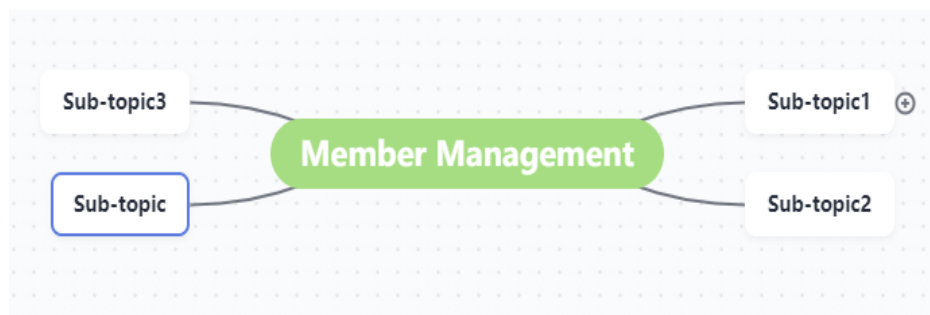
- Step 1** Enter a mind map. In the upper right corner of the map, click  on the left of the search box.
- Step 2** In the search box, enter a node name.
- Step 3** In the text box below the search box, enter a new name.
- Step 4** Click  to replace the name of a node with the new name. The next node to be renamed is marked in a blue box. After all nodes are renamed, a message is displayed indicating that all the nodes have been renamed.

Click  to rename nodes in batches. A progress dialog box is displayed on the page.

----End

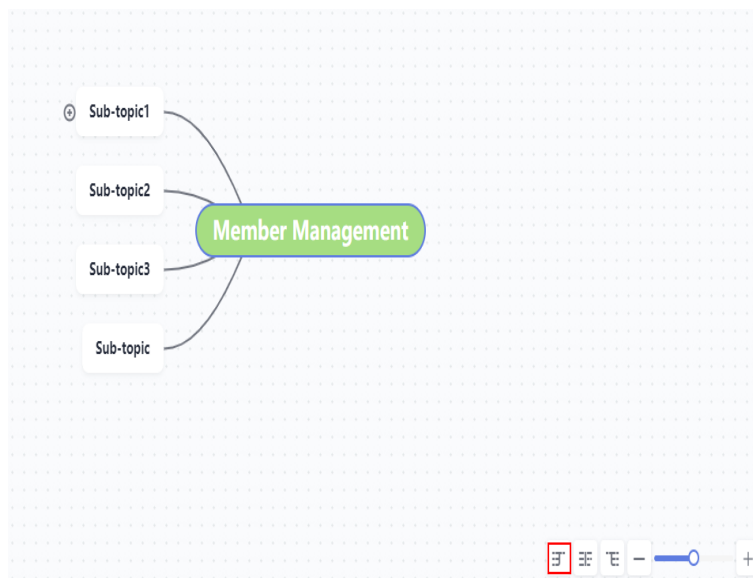
Adjusting Layout

By default, the mind map is in the center view. That is, the root node is in the center, and subnodes are distributed on both sides of the root node.

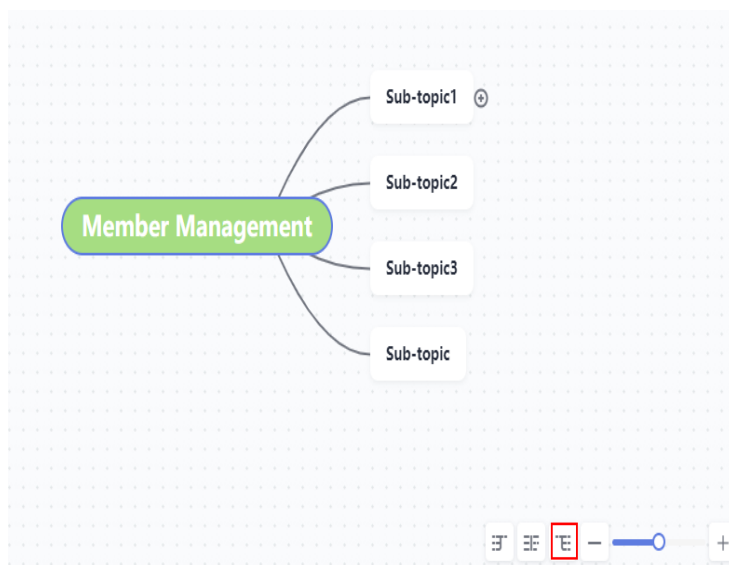


Click the toolbar in the lower right corner of the mind map to adjust the layout of the mind map.

- Click  to adjust the subnodes to the left of the root node.



- Click  to adjust the subnodes to the right of the root node.



- Click  to adjust the view to the default center view.
- Click  or  to zoom out or zoom in the mind map.

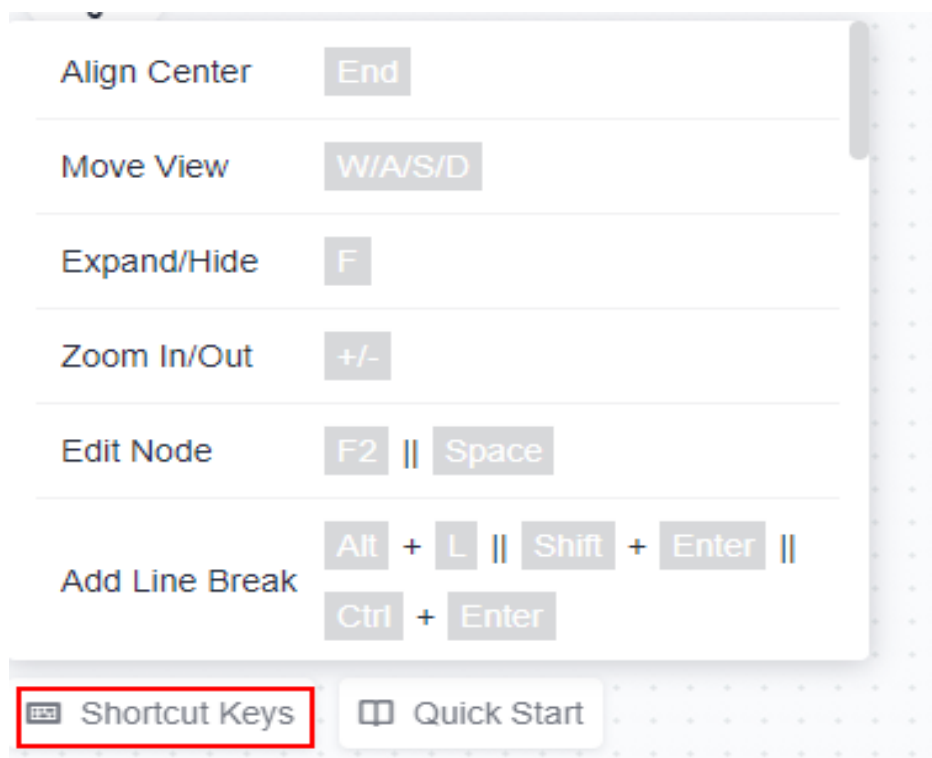
Shortcut Keys

You can use the following shortcut keys to edit the mind map during test design.

Operation	Shortcut Key
Align Center	End
Move View	W/A/S/D
Expand/Hide	F

Operation	Shortcut Key
Zoom In/Out	+/-
Edit Node	F2 or Space
Add Line Break	Alt+L/Shift+Enter/Ctrl+Enter
Copy	Ctrl+C
Paste	Ctrl+V
Cut	Ctrl+X
Undo	Ctrl+Z
Redo	Ctrl+Y
Add Sibling	Enter
Adding Child	Ins or Tab
Remove Node	Del
Select Root Node	Ctrl+Home or Home
Select Parent Node	Backspace
Move Selected Node	↑/←/↓/→
Move Node	Ctrl+↑/←/↓/→
Add Scenario	Alt+C
Add Test Node	Alt+P
Add Prerequisite	Alt+O
Add Step	Alt+T
Add Expected Result	Alt+X
Add Action Factor	Alt+Shift+A
Add Data Factor	Alt+Shift+D
Add Case Level	Ctrl+0 / 1 / 2 / 3 / 4
Find/Replace	Ctrl+F
Add Image	Ctrl+I
Add File	Ctrl+D
Set Tag	F3
View Shortcut Keys	Ctrl+Shift+L

Click **Shortcut Keys** in the lower left corner of the mind map to view the shortcut key list.



Copying a Mind Map in a Project or to Other Projects

Copy and paste a mind map within the same project or to other projects.

Step 1 Go to the **Testing Design** page.

Step 2 Locate the row of the mind map to be copied, click **...** in the **Operation** column, and click **Copy**.

To copy multiple mind maps, select them and click **Batch Copy** in the tool bar below the list. Up to 10 mind maps can be copied each time.

Step 3 In the mind map list page of the current project or other project, click **Paste** in the upper right corner of the list.

----End

Changing Mind Map Creator

Step 1 Go to the **Testing Design** page.

Step 2 Locate the target mind map and click **...** in the **Operation** column.

Step 3 Click **Change Permission**.

Step 4 In the displayed dialog box, click the drop-down list and select the project member to whom you want to transfer the mind map permissions.

Step 5 Click **OK**.

----End

6.3 Creating a Manual Test Case

Manual test cases are designed by testers based on requirements or product features to manage test scenarios and steps, and to verify whether software can function as expected.

Prerequisites

You have been assigned a role (except the O&M manager, viewer, and participant) that has permissions to create test cases. For details, see [User roles and permissions](#).

Notes and Constraints

For Scrum and IPD projects, work items that can be added to manual test cases as requirements are epics, features, and stories.

For Kanban projects, the default requirement work items can be added to manual test cases as requirements.

Creating a Manual Test Case

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** **Baseline** and **Test case library** are displayed by default. On the **Manual Test** tab page, click **Create Case** on the left.
- Step 4** Enter the name, description, prerequisites, and test steps as required, and click **Save**.

In the **Basic Information** area, click **Manage Field** to open the test field settings page. For details, see [Setting Test Case Fields](#).

The following table describes the main configuration items of manual test cases.

Configur ation Item	Description
Name	(Mandatory) Enter 1 to 128 characters. Describe the test scenario or function.
Descriptio n	Brief description of a test case. Enter the test scope, test objectives, test policy, test methods, test tools, test data, test metrics, and test environments of the current test case.
Prerequisi tes	Prerequisites for executing the current test case.

Configuration Item	Description
Test Steps	Step description and expected result.  : Add a row.  : Copy the row and paste it to the next row, or move it down.  : Delete the row.
Tag	Set tags for the current test case as required. Separate tags by spaces or carriage returns. Each test case can be associated with a maximum of 10 tags.
ID	Test case ID. You can enter a custom ID. If left blank, it will be automatically generated. 1. In the navigation pane, choose Settings > Testing. 2. Click the Test Case Settings tab, locate the row of ID, and click  in the Operation column. 3. In the displayed dialog box, enter the prefix, auto-increment start value, and connector. 4. Click OK. Test cases will be automatically numbered by this rule. The test case ID can be changed as required at the test case list page. An ID can contain 3 to 128 characters.
Case Level	Test case level based on the importance of the scenario or function. The default level is L2. • L0: verification of underlying functions. Each module should have 10 to 20 test cases. L0 test cases account for 5% of all test cases. • L1: verification of basic functions for inherited features or before sprint acceptance. L1 test cases account for 20% of all test cases. • L2: verification of important features for manual tests in non-regression versions. L2 test cases account for 60% of all test cases. • L3: verification of minor and non-important functions, and exception tests on basic and important functions. L3 test cases account for 15% to 20% of all test cases. • L4: verification of special input, scenarios, and threshold conditions. L4 test cases account for less than 5% of all cases.
Module	Module of the current test case. The module list comes from the project settings. For details, see Adding Work Item Modules.
Version	Version number of the current test case.
Sprint	Sprint to test the current test case.
Processor	Person who needs to complete the test task.

Configuration Item	Description
Associated Requirement	It is recommended that test cases be bidirectionally associated with test requirements. If the current test case is used to test a specific requirement and the requirement has been recorded in the CodeArts Req service, click Associate to associate the test case with the requirement work item.
Folder	Associated test feature directory. The test case is in the Other directory by default.
Attachment	Attachment related to the current test case. You can associate an existing file from the Files page or upload an attachment from the local PC.
Custom Field	Customized fields set in the Settings page. For details, see Setting Test Case Fields .

----End

Editing Test Case Details

Take a Scrum project as an example. Related requirements have been recorded in the project management module for the current sprint.

Before testing, the tester creates a manual test case for the corresponding function in the test case library based on front-end requirements, and associates them.

- Step 1** After the operations described in [Creating a Manual Test Case](#) are complete, choose **Testing Design > Manual Test**, click the name of the test case to be edited, and click the **Details** tab.
- Step 2** Edit the test details. Two views are provided for editing test steps:
- Text view. Click **Text** on the segmented button and enter test steps and expected results in the two text boxes.
 - Table view. Click **Table** on the segmented button and enter test steps and expected results in the table. To add more rows, click **+** in the **Operation** column.
- Step 3** Click the **Requirements** tab. Click **Associate With Requirement** on the right of the page. In the displayed dialog box, select the requirements to be associated and click **OK**.
- Step 4** Click **Save**.

----End

Helpful Links

- [Adding Test Cases to a Test Plan](#) and [Creating a Manual Test Suite](#): Add test cases to test plans and suites to assign execution tasks to testers.

- **Managing Test Cases:** Import and export manual test cases and associate requirements or bugs.

6.4 Creating an Automated API Test Case

6.4.1 Using Automated API Test Cases

A rich UI is provided for writing of automated API test cases. You can orchestrate test steps in a visual manner, use various checkpoints, configure extraction as needed, and quickly generate test cases that match the core API logic without coding. You can arrange API test sequence by dragging and dropping test steps to generate your own strategies for automated testing. You can also generate automated API test scripts from scenarios to avoid repetitive tasks, and focus on exploratory testing and other creative and valuable test activities.

An automated API test case consists of basic information and scripts.

- Basic information is for managing and describing the test case, including the name (mandatory), type, module, version, sprint, associated requirements, ID, tag, test case level, processor, folder, description, prerequisites, test steps, and expected results.
- A script defines automated test procedure, including test steps, logic control, and test parameters.

The process of using an automated API test case consists of three stages: preparation, testing, and destruction.

The preparation stage corresponds to the **Pre-steps** tab page to prepare for the test prerequisites. The testing stage corresponds to the **Test Procedure** tab page to implement the API function test. The destruction stage corresponds to the **Post-steps** tab page to release or restore the test data in the preparation and testing stages.

- (Optional) Preparation stage
 - In this stage, prepare the prerequisite data required in the testing stage. If there is no prerequisite, skip this stage.
 - In this stage, the prerequisites are initialized using API calls. If the data of the prerequisites needs to be referenced in the testing stage, you can use parameter passing to parameterize the data. For details, see [Setting the Response Extraction of an API Script](#).
- Testing stage

Define the API core test steps. The test steps in the testing stage can reference the parameters extracted in the preparation stage.
- (Optional) Destruction stage
 - To avoid impacting other tests or the following test, clear the test environment data after each test, reset the test environment, and delete the data created in the preparation stage.
 - If no data needs to be deleted, ignore this stage. You can delete data using API calls. The test steps in the destruction stage can reference the parameters extracted in the preparation stage.

6.4.2 Creating an Automated API Test Case Template

Notes and Constraints

The name of an API automation test case must contain 1 to 128 characters. Only letters, digits, and special characters (-_/*&`'^~;:(){}=+,x...—!@#\$.[]<>?-"') are supported.

Creating an Automated API Test Case

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** Click the **Auto API Test** tab and click **Create Case** in the upper left corner of the page.
- Step 4** Enter the test case name, set other information as required, and click **Save**. Alternatively, click **Save and Write Script**. The **Scripting** page is displayed.

The main configuration items of automated API test cases are similar to those of manual test cases. For details, see [Creating a Manual Test Case](#).

----End

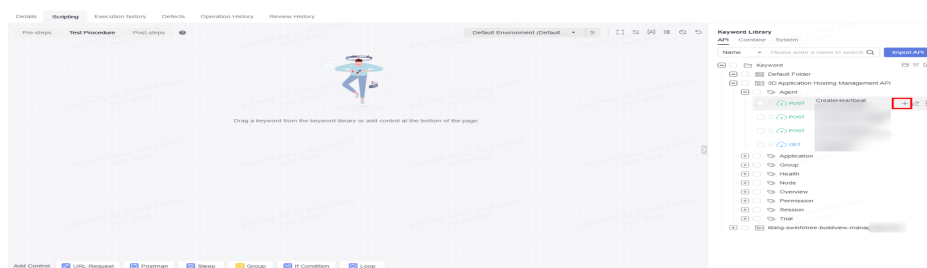
6.4.3 Adding an API Test Script by Using a Custom URL Request

6.4.3.1 Adding and Setting the URL Request of an API Script

- Step 1** After the operations described in [Creating an Automated API Test Case Template](#) are complete, choose **Testing Case > Auto API Test** and click the name of the test case to be edited.
- Step 2** Choose **Scripting > URL Request** to generate a test step.

You can import a Swagger description file of the API to be tested to generate a script template and then orchestrate test cases. For details, see [Saving Test Procedure as an API Keyword](#).

Select a script template and drag and drop the script template card or click **+** on the card to add the script to the **Test Procedure** tab page.



- Step 3** Edit the URL request as required by referring to [Adding and Setting the URL Request of an API Script](#), [Setting Checkpoints for an API Script](#), and [Setting the Response Extraction of an API Script](#). Enter a domain name or IP address as the environment address in the request. If you import a Swagger or Postman file, the address will be automatically generated to the request.
- Step 4** (Optional) Repeat [Step 2](#) to [Step 3](#) to add pre-steps and post-steps.
- Step 5** After the editing is complete, click **Save** in the upper right corner of the page to save the script.

----End

 NOTE

- Automated API test cases support the use of built-in functions in the request URL path, request header, request body, checkpoint parameters, and URL response. For details about built-in functions, see [Built-in Functions](#).
- When editing automated API test cases, you can right-click the title area of a test step to cut, copy, paste, and delete a test step. If there are multiple test steps, you can press **Ctrl+left-click** to select multiple test steps and right-click them in batches. After a test step is cut or copied, the test step can be pasted on the current tab page, across tab pages, or across test cases.
- Note that the right-click response in the title area is the operation on the test step. So, when you edit the text box in the title area of the test step, the shortcut menu of the browser is overwritten by the shortcut menu of the system, and the shortcut menu of the browser does not take effect. To copy and paste text in the text box of the test step title area, press **Ctrl+C** and **Ctrl+V**.

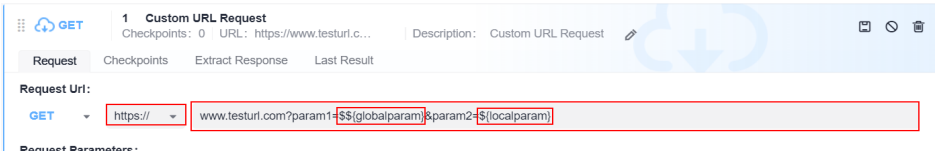
Requesting URL and URL Parameters

On the **Scripting** tab page of the automated API test cases, enter the URL to be requested. HTTP and HTTPS requests are supported.

API automation supports the following URL request modes. By default, the request mode of a new URL is **GET**.

Request Mode	Description
GET	Retrieves data from APIs.
POST	Uploads files or images and adds data.
PUT	Replaces the existing data.
DELETE	Deletes the existing data.
HEAD	Obtains the HTTP header of a response.
OPTIONS	Pre-checks requests.
PATCH	Updates the fields of some existing data.

The request URL supports environment parameters, local parameters, and response extraction parameters. For details, see [Setting Test Case Parameters of an API Script](#).



Setting Request Header in a URL Request

Common HTTP request headers are preset for API automation. In the request header module, enter the request header information.

- Step 1
- In the API automation test case script, **add a URL request**.
- Step 2
- Click the URL request name.
- Step 3
- Set the request header for the current URL request.

The request header can be in a form or text. By default, the form mode is used on the page. You can click   to switch between the form and text modes.

Header: 

Name	Value	Description
Content-Type	application/json	
Accept	application/json;charset=UTF-8	

+ Q

- Form: Select or enter a request header name in the **Name** column, and select or enter a value in the **Value** column.
- Text: The request header must be in the format of **key:value**.
Unlike the form mode, the text mode supports only the configuration of **Request Header** and **Value**, but not **Description**.
Pay attention to the following restrictions when setting request headers:
 - The total number of request headers cannot exceed 30.
 - The length of the request header cannot exceed 10000.

Header:

Content-Type:application/json
Accept:application/json;charset=UTF-8

----End

Multiple common HTTP request headers are preset in CodeArts TestPlan. For details, see the following table.

Table 6-1 Request header types

Request Header	Description
Accept	Acceptable response content type (Content-Types), for example, text/plain.
Accept-Charset	Acceptable character set, for example, UTF-8.
Accept-Encoding	List of acceptable encoding modes, for example, compress gzip identity.
Accept-Language	Natural language list of acceptable response content, for example, en-US.
Accept-Datetime	Acceptable versions displayed by time.
Access-Control-Request-Method	Pre-checks requests so that the server knows which HTTP methods will be used when the actual requests are sent.
Access-Control-Request-Headers	Pre-checks requests so that the server knows which HTTP headers will be used when the actual requests are sent.
Authorization	Information used for HTTP authentication.
Cache-Control	Instructions that must be followed by all cache mechanisms for a request or response chain.
Connection	Preferred connection type used by the browser.
Cookie	HTTP cookie sent by servers through the Set-Cookie.
Content-Length	Length of the request body represented by 8-byte arrays.
Content-MD5	Binary MD5 hash value of the content of the request body, which is encoded using Base64.
Content-Type	Multimedia types of the request body (used in POST and PUT requests), for example, application/json.
Date	Date and time when a message is sent.
Expect	Specific actions required by a client for a server.
Forwarded	Client-oriented information from the path proxy server that the proxy participates in the request when it is changed or lost. It is used for debugging, collecting statistics on, and generating location-dependent content, and is designed to display privacy-sensitive information, such as the IP address of the client. So, pay attention to user privacy when deploying this header.

Request Header	Description
From	Email address of the user who initiates the request.
Host	Domain name of the server (used for the virtual host) and the port number of the transmission control protocol listened to by the server. If the requested port is the standard port of the corresponding service, the port number can be omitted. This field is mandatory since HTTP/1.1. If the domain name in the URL is an IP address, this field is automatically added. Otherwise, enter the IP address and port number of the tested application in this field.
If-Match	The corresponding operation is performed only when the entity provided by the client matches the entity on the server. It is used in a method such as PUT to update a resource which has not been modified since the last update.
If-Modified-Since	Return 304 Not Modified is allowed when the corresponding content is not modified.
If-None-Match	Return 304 Not Modified is allowed when the corresponding content is not modified. Refer to the HTTP entity tag. In a typical use, when a URL is requested, the web server returns the resource and its corresponding ETag value, which is placed in the ETag field of the HTTP. The client can then decide whether to cache the resource and its ETag. If the client requests the same URL in the future, a request containing the saved ETag and the If-None-Match field is sent.
If-Range	If an entity is not modified, one or more parts that are missing are sent to the sender. Otherwise, the entire new entity is sent.
If-Unmodified-Since	A response is sent only when the entity has not been modified since a specific time.
Max-Forwards	Number of times a message can be forwarded by the proxy and gateway.
Origin	Initiates a request for cross-origin resource sharing. The server is mandatory to add an Access-Control-Allow-Origin field to the response.
Pragma	Related to specific implementations and may produce multiple effects at any time in the request or response chain.
Proxy-Authorization	Information used to authenticate a proxy.
Range	Requests only a part of an entity with the byte offset starting from 0.

Request Header	Description
Referer	Previous page accessed by a browser. A link on this page brings the browser to the currently requested page.
TE	Transmission coding mode expected by a browser. You can use a value of Transfer-Encoding in the response protocol header. In addition, the value trailers (related to the block transmission mode) indicates that the browser expects to receive additional fields if the size of the last block is 0.
User-Agent	String of the browser identity.
Upgrade	Asks the server to be upgraded to another protocol.
Via	Request-sending agents informed to the server.
Warning	General warning indicating that errors may exist in the body of an entity.

Setting a General Request Header in the Script of an API Automation Test Case

Configure a general request header for reuse by subsequent test steps in the script to improve test efficiency.

Step 1 Go to the **Scripting** tab page.

Step 2 Click **Header** at the bottom of the page.

Step 3 A request header step is generated on the script page.

Step 4 Click  in the request header to add a description (a maximum of 100 characters) to the request header control.

Step 5 Set the value (using a local parameter, environment parameter, built-in function, or standard parameter). For details about the request header types, see [Table 6-1](#). A maximum of 30 values can be added, and the value length cannot exceed 10,000 characters.

Step 6 Drag the header above any test step. The steps below the header inherit its values.

----End

NOTE

- If the general request header is set as the first test step, it takes effect for all subsequent requests.
- If a general request header shares the same name as a URL request header, the URL request header takes precedence.
- A general request header can be combined with other steps to form keywords. For details, see [Saving Test Procedure as a Combined Keyword](#).

Request Body

A request body is a message (packet) to be transferred in an API request. The request body can be in text, JSON, or form format.

If the request method is **POST**, **PUT**, **DELETE**, **OPTIONS**, **PATCH**, or **HEAD**, the request body is displayed on the page. If the request method is **GET**, the request body is not displayed.

- Text: You can enter a standard JSON string. The method is the same as that for selecting a JSON request body.

Request Body:

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw

JSON

1

Text
JSON

- Form-data: Both text and file types are supported. You can add up to 100 form-data request headers.
 - Text: Set the parameter name and value.

Request Body:

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw

Type	Name	Value
Text	date	2023-3-30

- File: Set the parameter name and assign a value to the parameter by uploading a file in any format.

Request Body:

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw

Type	Name	Value
File		--select--

6.4.3.2 Setting Checkpoints for an API Script

Suggestions

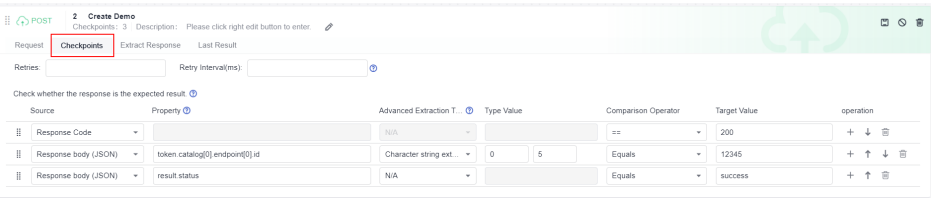
You are advised to set checkpoints. For API requests, provide checkpoints for determining response codes.

If the checkpoints are left empty, the result is success regardless of the response code of the API.

Checkpoint Description

A test checkpoint is also called assertion. It determines whether the system meets the expectation based on the API response.

On the **Checkpoints** tab page of the test step in the test case details of API automation, you can define test checkpoints.



Checkpoint parameters include **Property**, **Comparison Operator**, and **Target Value**.

Parameter	Description
Retries	Number of times that the test step is re-executed if the checkpoint fails. The value is an integer ranging from 0 to 5. After you set the value, the response for every retry attempt (including failed or abnormal requests) appears in the execution history.
Retry Interval	Interval between retries if the checkpoint fails. The unit is ms. The value is an integer ranging from 0 to 10,000.
Source	Source of the detected field, such as the response body (JSON), response header, response code, and variable.

Parameter	Description
Property	Type \$ to reference global variables, local variables, or built-in functions. • If the source is a response code, you can leave this parameter blank. For details, see Response Code Check. • If the source is a response header, specify the field name in the response header for this property. For details, see Response Header Check. • If the source is the JSON response body, set the property using one of the following methods: 1. Standard extraction expression (does not start with \$), for example, item.name. Obtain the value of a field. Nested values are supported. For details, see Response Body (JSON) Check. When an array is extracted from the response body, the index can be a number or a key:value expression. For details, see Example: Obtaining a String from the Response Body Based on the Specified key:value. 2. JSONPath expression (starts with \$. or \$[]), for example, \$.store.book[0].title. For details, see Obtaining Data from the Response Body Based on JSONPath. • If the source is a variable, specify a global variable, local variable, or extracted response variable. For details, see Variables Check.
Advanced Extraction Type	(Optional) Use the advanced extraction type to assist in extracting checkpoint information. If N/A is selected, no additional matching mode is used. Currently, there are two modes: • Character string extraction (truncation). For details, see Character String Extraction Description. • Regular expressions to filter source strings. For details, see Regular Expression Description. For the advanced extraction type, the string extraction function is preferred. If the function cannot meet the requirements, you can use the regular expression.
Type Value	Parameter required in the advanced extraction type.
Comparison Operator	Comparison operator, such as numbers, strings, JSON objects, and types, are supported. For details, see Comparison Operator Description .
Target Value	Expected value of a checkpoint. Built-in parameters can be used for target values. For details, see Built-in Parameters .

For example, the following table describes the parameter settings to check whether the range from the 0th digit (first digit) to the 4th digit of the **item.name** field in the response body (JSON format) is **petty**.

Parameter	Value
Source	Response body (JSON)
Property	item.name
Advanced Extraction Type	Character string extraction
Type Value	0-5
Comparison Operator	Equals (string)
Target Value	petty

Character String Extraction Description

When setting checkpoints or response extraction, if the expressions in the property setting bar cannot meet the requirements, you can set **Advanced Extraction Type** to **Character string extraction**.

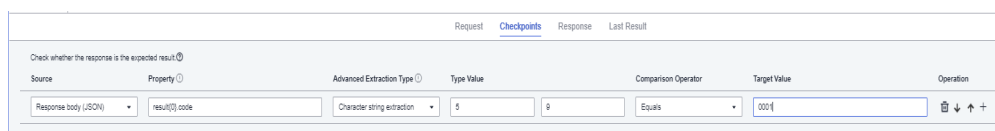
There are two **Type Value** text boxes for string truncation:

- The first box indicates the start index. The first digit is **0**. The start index is contained in the truncated string.
- The second box indicates the end index. The end index is not contained in the truncated string.

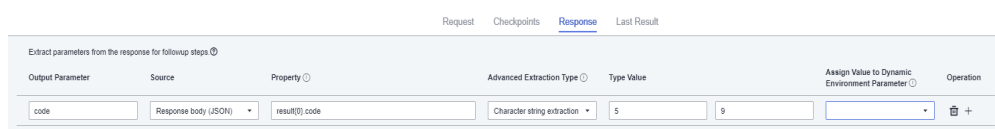
The following response body is used as an example.



- To extract the fifth to ninth digits of **code** in the first element of the **result** array from the response body and compare them with the target value, configure the checkpoints by referring to the following figure.



- To extract the fifth to ninth digits of **code** in the first element of the **result** array in the response body and assign the value to the variable **code**, configure the response extraction by referring to the following figure.

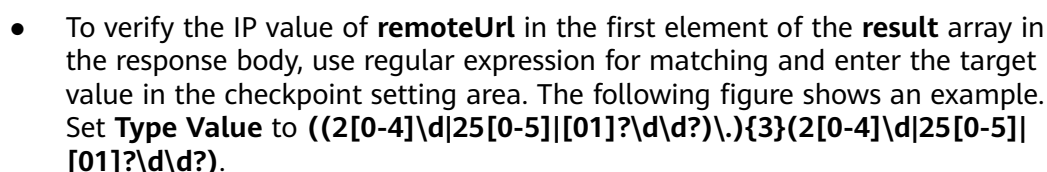


Regular Expression Description

When setting checkpoints or response extraction, if the expressions in the property setting bar or the character string extraction function cannot meet the requirements, you can set **Advanced Extraction Type** to **Regular expression**.

The Java regular expression engine can be used.

The following response body is used as an example.



- To extract the IP value of **remoteUrl** in the first element of the **result** array in the response body, and assign the value to the variable **remoteIp**, use regular expression for matching and set the response extraction by referring to the following figure. Set **Type Value** to **((2[0-4]\d|25[0-5])|[01]?\d\d?)\.){3}(2[0-4]\d|25[0-5])|[01]?\d\d?)**.

66

Comparison Operator Description

CodeArts TestPlan supports the following comparison types.

Comparison Type	Comparison Operator	Value Mandatory or Not	Example
Digit	• == • != • >= • <= • > • <	Yes	• Response code = 200 • Response code != 200 • Response code >= 200 • Response code <= 200 • Response code > 200 • Response code < 200
String	• Equals • Does not equal • Equals (case insensitive) • Contains • Does not contain	Yes	• Parameter 1 in the response body equals test. • Parameter 2 in the response body does not equal test. • Parameter 3 in the response body equals TEST. • Parameter 4 in the response body contains tri. • Parameter 5 in the response body does not contain tri.
Regular expression	Regular expression	Yes	• Regular expression for parameter 1 in the response body: ^[A-Za-z0-9]{1,32}\$
General	• Is empty • Is not empty	No	• Parameter 1 in the response body is empty. • Parameter 2 in the response body is not empty.
JSON array	• Has null JSON array • Has non-null JSON array	No	• Parameter 1 in the response body has a null JSON array. • Parameter 2 in the response body has a non-null JSON array.

Comparison Type	Comparison Operator	Value Mandatory or Not	Example
	JSON array size	Yes	• Size of the parameter 1 JSON array in the response body.
Type	- Is JSON type - Is JSON array type	No	- Parameter 1 in the response body is of the JSON type. - Parameter 2 in the response body is of the JSON array type.
JSON object	JSON value	Yes	• The JSON value of parameter 1 in the response body is {"name":"zhangsan"} .

Response Code Check

Compare the response code with the target value. For example, check whether the response code is **200**.

The screenshot shows the 'Checkpoints' tab in the CodeArts TestPlan interface. It displays a configuration for a response code check. The 'Retries' field is set to 0, and the 'Retry Interval(ms)' is set to 0. Below this, there is a section for configuring the check: 'Check whether the response is the expected result.' The 'Source' is set to 'Response Code', the 'Property' is 'N/A', the 'Advanced Extract...' is 'N/A', the 'Type Value' is 'N/A', the 'Comparison Operator' is 'Equals', the 'Target Value' is '200', and the 'operation' is '+'. There are also icons for adding and deleting checkpoints.

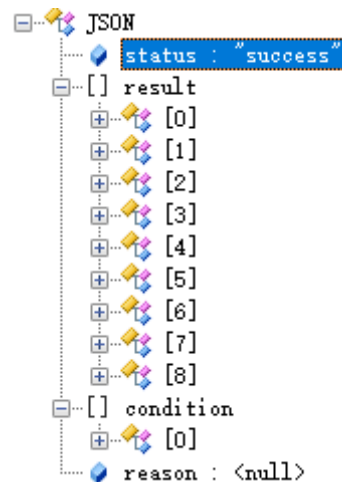
Response Header Check

Compare the value of a field in the response header with the target value. For example, check whether the value of **content-type** in the response header is equal to **text/plain;charset=UTF-8**.

The screenshot shows the 'Checkpoints' tab in the CodeArts TestPlan interface. It displays a configuration for a response header check. The 'Retries' field is set to 0, and the 'Retry Interval(ms)' is set to 0. Below this, there is a section for configuring the check: 'Check whether the response is the expected result.' The 'Source' is set to 'Response Header', the 'Property' is 'content-type', the 'Advanced Extract...' is 'N/A', the 'Type Value' is 'N/A', the 'Comparison Operator' is 'Equals', the 'Target Value' is 'text/plain;charset=UTF-8', and the 'operation' is '+'. There are also icons for adding and deleting checkpoints.

Response Body (JSON) Check

1. Check the value of the object field in the response body (JSON). For example:
The response body structure is as follows.

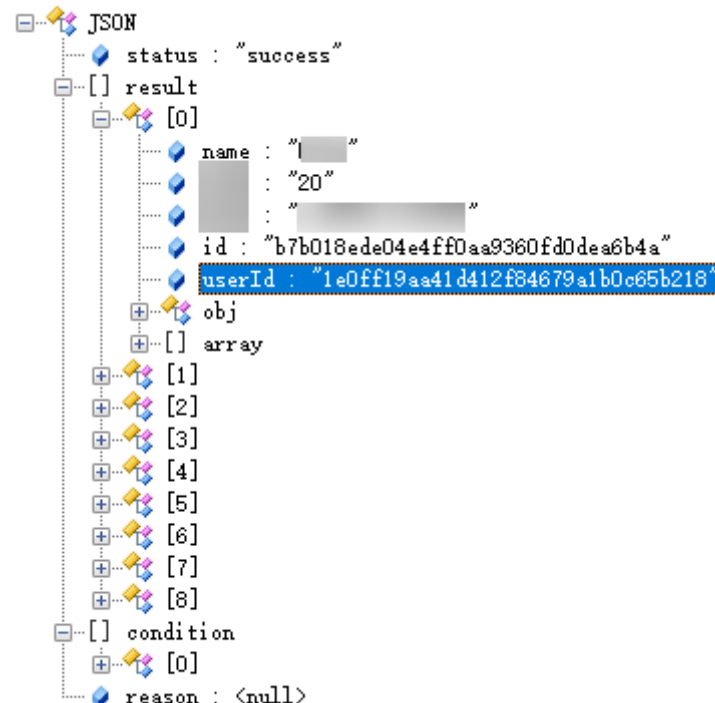


To check the value of the field whose name is **status** in the response body object, configure the checkpoint by referring to the following figure.

The checkpoint configuration interface shows the "Checkpoints" tab selected. The "Retries" field is set to 1 and the "Retry Interval(ms)" field is set to 1000. The "Check whether the response is the expected result." checkbox is checked. The "Source" dropdown is set to "Response body (...)", the "Property" dropdown is set to "status", the "Advanced Extrac..." dropdown is set to "N/A", the "Type Value" dropdown is set to "Text", the "Comparison Operator" dropdown is set to "Equals", and the "Target Value" field is set to "success".

2. Check the field value of an array object in the response body (JSON). (The array condition uses the subscript starting from 0 to determine the object.)
For example:

The response body structure is as follows.



To check the value of **userId** in the first element object field of the **result** array in the response body, configure the checkpoint by referring to the following figure.

Request

Checkpoints

Extract Response

Last Result

Retries: Retry Interval(ms):

Check whether the response is the expected result.

Source	Property	Advanced Extraction Type	Type Value	Comparison Operator	Target Value	operation
Response body (JSON)	result.userId	N/A		Equals	3c3788524e454265b016929c2932f891	+ -

3. Check the field value of an array object in the response body (JSON). (The array condition uses the fuzzy match function to determine the object.) For example:

The response body structure is as follows.



- Check all objects whose **name** is **beer** in the **result** array of the response body. Obtain the values of **id** after the first object. Configure the checkpoint by referring to the following figure.

The screenshot shows the 'Checkpoints' tab in the test case configuration. It displays a table with one checkpoint. The 'Source' is 'Response body (JSON)', the 'Property' is 'result[name==beer][0].id', the 'Advanced Extraction Type' is 'N/A', the 'Type Value' is empty, the 'Comparison Operator' is 'Equals', and the 'Target Value' is '3c3788524c454260f18929e283298f1'. There are also fields for 'Retries' and 'Retry Interval(ms)'.

NOTE

If there is only one object in the obtained array, **[0]** can be omitted. The expression in the example can be written as **result[name==beer].id**.

- b. Check the ID of the object whose **name** in the **result** array of the response body is **beer**, and of which the **a** property under **obj** is **2**. Configure the checkpoint by referring to the following figure.

The screenshot shows the 'Checkpoints' tab with a single checkpoint. The 'Source' is 'Response body (JSON)', the 'Property' is 'result[name==beer&&obj.a==2].id', the 'Advanced Extraction Type' is 'N/A', the 'Type Value' is empty, the 'Comparison Operator' is 'Equals', and the 'Target Value' is '3c3788524c454260f18929e283298f1'.

Variables Check

Check global variables, local variables, and variable values after response extraction, and compare them with target values. For example:

- Check whether the value of the global variable **hostIp** is **127.0.0.1**.
- Check whether the value of the local variable **local** is **test**.
- Check whether the value of the variable **name** after response extraction is **Liquor**.

The screenshot shows the 'Checkpoints' tab with three checkpoints. Each has 'Source' as 'Variable' and 'Property' as a variable name. The first has 'Property' '\$(hostURL)' and 'Target Value' '127.0.0.1'. The second has 'Property' '\$(local)' and 'Target Value' 'test'. The third has 'Property' '\$(name)' and 'Target Value' 'Liquor'. All have 'Comparison Operator' set to 'Equals'.

Example: Obtaining a String from the Response Body Based on the Specified key:value

You can enter an array whose subscript is in the **key:value** format for the **Property** field of the check body to obtain the JSON string that meets the condition from the response body based on the specified **key:value**.

The following connectors are supported between **key:value**.

Connector	Description	Example
==	Equals (string or digit)	name==John age==20
!=	Does not equal (string or digit)	name!=John age!=20
>	Greater than (digit)	age>20
>=	Greater than or equal to (digit)	age>=20
<	Less than (digit)	age<20

Connector	Description	Example
<=	Less than or equal to (digit)	age<=20

The array subscript supports multiple groups of **key:value** conditions connected by **&&**, indicating that the extracted JSON string must meet all **key:value** conditions. For example, **student[age>20&gender=female]** indicates that the extraction conditions are that the age must be greater than 20 and the gender must be female.

The search criteria of an array support nested arrays. In this case, the sub-condition can be met only when all objects in the nested condition meet the conditions.

NOTE

- If the value is an empty object, use **\$null**, that is, **key==\$null**.
- If the value is a null string ("null"), use **null**, that is, **key==null**.
- If the value is an empty string (""), use **key==**.
- If the property value of a checkpoint contains special characters, use single quotation marks or double quotation marks as boundary characters.

For example, **name=="zhangsan(ab)"**.

The following response body (JSON) is used as an example:

```
{
  "status": "success",
  "result": [
    {
      "name": "Beer",
      "quantity": "20",
      "address": "Shelf 10 in repo A3",
      "obj": {
        "a": 1,
        "b": "test",
        "c": "test"
      },
      "array": [
        {
          "id": 1,
          "name": "aaa"
        },
        {
          "id": 2,
          "name": "bbb"
        }
      ]
    },
    {
      "name": "Beer",
      "quantity": "10",
      "address": "Shelf 10 in repo A3",
      "obj": {
        "a": 1,
        "b": "test",
        "c": "test"
      },
      "array": [
        {
          "id": 1,
          "name": "aaa"
        }
      ]
    }
  ]
}
```

```
    },
    {
      "id": 2,
      "name": "bbb"
    }
  ]
},
{
  "name": "Liquor",
  "quantity": "20",
  "address": "Shelf 10 in repo A3",
  "obj": {
    "a": 1,
    "b": "test",
    "c": "test"
  },
  "array": [
    {
      "id": 1,
      "name": "aaa"
    },
    {
      "id": 2,
      "name": "bbb"
    }
  ]
},
{
  "name": "Liquor",
  "quantity": "30",
  "address": "Shelf 10 in repo A3",
  "obj": {
    "a": 1,
    "b": "test",
    "c": "test"
  },
  "array": [
    {
      "id": 3,
      "name": "aaa"
    },
    {
      "id": 4,
      "name": "bbb"
    }
  ]
},
{
  "name": null,
  "quantity": "10",
  "address": "Shelf 10 in repo A3",
  "obj": {
    "a": 2,
    "b": "test",
    "c": "test"
  },
  "array": [
    {
      "id": 5,
      "name": "aaa"
    },
    {
      "id": 6,
      "name": "bbb"
    }
  ]
},
{
  "name": "",
```

```
{
  "quantity": "10",
  "address": "Shelf 10 in repo A3",
  "obj": {
    "a": 2,
    "b": "test",
    "c": "test"
  },
  "array": [
    {
      "id": 5,
      "name": "aaa"
    },
    {
      "id": 6,
      "name": "bbb"
    }
  ]
},
"condition": [
  {
    "name": "Beer",
    "quantity": "120",
    "address": "Shelf 15 in warehouse A1",
  }
],
"reason": null
}
```

The following table lists the expressions for obtaining data from the response body.

Extraction Condition	Expression
Obtain the data whose name is Liquor from the result array.	result[name==Liquor]
Obtain the data whose name is not Beer and quantity is greater than 20 from the result array.	result[name!=Beer&&quantity>20]
Obtain the data whose name is a null object and id is less than or equal to 2 from the result array.	result[name==\$null&&array[id<=2]]
Obtain the data whose name is a null string and property a under obj is 2 from the result array.	result[name==null&&obj.a==2]
Obtain the data whose name is an empty string ("") from the result array.	result[name==]

If you need to check whether the data (JSON) whose **name** is **Liquor** and **quantity** is greater than 20 is [{"name": "Liquor", "quantity": "30", "address": "Shelf 10 in warehouse A3", "obj": {"a": 1, "b": "test", "c": "test"}, "array": [{"id": 3, "name": "aaa"}, {"id": 4, "name": "bbb"}]}, refer to the following table for the configuration of each field.

Parameter	Value
Source	Response body (JSON)
Property	result[name!=Beer&&quantity>20]
Comparison Operator	Equals (string)
Target Value	[{"name": "Liquor", "quantity": "30", "address": "Shelf 10 in warehouse A3", "obj": {"a": 1, "b": "test", "c": "test"}, "array": [{"id": 3, "name": "aaa"}, {"id": 4, "name": "bbb"}]}

Example: Obtaining Data from the Response Body Based on JSONPath

For details about JSONPath, visit the [official website](#).

The following response body (JSON) is used as an example:

```
{
  "store": {
    "book": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99
      },
      {
        "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99
      },
      {
        "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95
    }
  },
  "expensive": 10
}
```

The JSONPath expression can contain dots (.) or brackets ([]). For example:

- \$.store.book[0].title

- `['store']['book'][0]['title']`

The following table describes the JSONPath expressions and their meanings.

JSONPath Expression	Description
<code>\$.store.book[*].author</code>	The authors of all books.
<code>\$..author</code>	All authors.
<code>\$.store.*</code>	All things, both books and bicycles.
<code>\$.store..price</code>	The price of everything.
<code>\$..book[2]</code>	The third book.
<code>\$..book[-2]</code>	The second to last book.
<code>\$..book[0,1]</code>	The first two books.
<code>\$..book[:2]</code>	All books from index 0 (inclusive) until index 2 (exclusive).
<code>\$..book[1:2]</code>	All books from index 1 (inclusive) until index 2 (exclusive).
<code>\$..book[-2:]</code>	Last two books.
<code>\$..book[2:]</code>	Book number two from tail.
<code>\$..book[?(@.isbn)]</code>	All books with an ISBN.
<code>\$.store.book[?(@.price < 10)]</code>	All books in store cheaper than 10.
<code>\$..book[?(@.price <= \$['expensive'])]</code>	All books in store that are not "expensive".
<code>\$..book[?(@.author =~ /*REES/i)]</code>	All books matching regex (ignore case).
<code>\$.*</code>	Give me everything.

NOTE

If the **length** or **size** function is used, do not use deep scanning (that is, the `..` symbol) when calling the function for multiple times. For example, if `$..book.length()` returns the number of books, determine the path and change it to `$.store.book.length()`.

6.4.3.3 Setting the Response Extraction of an API Script

Notes and Constraints

A maximum of 40 response extraction parameters are supported.

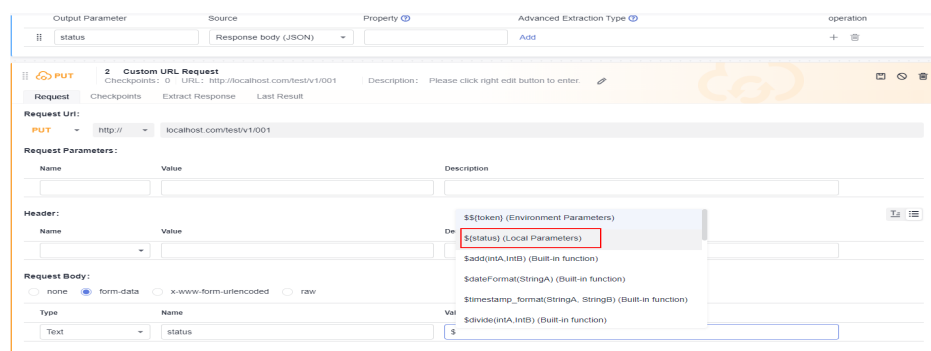
Function Description

Response extraction is to extract a part of the API response result and name it as a parameter for invoking in subsequent test steps. Response extraction needs to be defined in the previous test steps and used in subsequent test steps.

You can mask sensitive information in response parameters by referring to [Step 7](#).

- In the **Pre-steps** tab page, create the parameters to be passed on the **Extract Response** tab page. Built-in parameters are used for response extraction. For details, see [Built-in Parameters](#). Response extraction also supports regular expression matching and extracts the return value that matches the given regular expression.
- In subsequent test steps, use **`${parameter name}`** to reference the response extraction created in the previous test steps. This parameter can be referenced in the URL, request header, and request body in subsequent steps. If this parameter is referenced in the request body in JSON format, enclose the parameter with quotation marks. For example:

```
{
  id: "Test case ID"
  name:"${name}"
}
```



- The response extraction function can obtain strings based on the specified **key:value**. For details, see [Example: Obtaining a String from the Response Body Based on the Specified key:value](#).

Parameter	Description
Output Parameter	<code>\${Output Parameter}</code> , which will be referenced later. The name consists of letters, digits, and underscores (_).
Source	Source of the detected field, such as the response body (JSON), response header, and response code.

Parameter	Description
Property	Enter \$ to invoke global variables, local variables, and built-in functions. • If the source is a response code, this parameter can be left empty. For details, see Response Code Check. • If the source is a response header, the property is the name of the field in the response header. For details, see Response Header Check. • If the source is the response body (JSON), the property can be set in either of the following ways: 1. Common extraction expression (not starting with \$), for example, item.name. Obtain the value of a field. Nested values are supported. For details, see Response Body (JSON) Check. When an array is extracted from the response body, the index can be a number or a key:value expression. For details, see Example: Obtaining a String from the Response Body Based on the Specified key:value. 2. JSONPath expression (starting with \$. or \$[]), for example, \$.store.book[0].title. For details, see Example: Obtaining Data from the Response Body Based on JSONPath.
Advanced Extraction Type	(Optional) Use the advanced extraction type to assist in extracting response information. If N/A is selected, no additional matching mode is used. Currently, there are two modes: • Character string extraction (truncation). For details, see Character String Extraction Description. • Regular expressions to filter source strings. For details, see Regular Expression Description. For the advanced extraction type, the string extraction function is preferred. If the function cannot meet the requirements, you can use the regular expression.
Assign Value to Dynamic Environment Parameter	Assigns the value extracted from the response to the dynamic parameter so that the dynamic parameter can be referenced in subsequent tests.

6.4.3.4 Setting Authentication in an API Automation Test Case Script

To ensure only authorized users or clients access the target API, use authentication to verify identity and permissions before granting access. Key functions include:

- **Identity verification:** Verifies credentials (such as a username and password) to ensure requests originate from an authorized user.
- **Permission control:** Checks whether a user has the permission to access specific resources or perform specific operations.
- **API security:** Prevents unauthorized access and protects sensitive data.

Authentication Methods

Select one of the following built-in authentication methods in API automation test case scripts, and configure verification information based on API requirements.

- **No Auth:** No authentication is required.
- **Bearer Token:** Includes the token in the request header using the format **Authorization: Bearer <token>**.
- **API Key:** Includes a predefined key in a request header or parameter, which the server validates upon receipt.

Setting Authentication in an API Automation Test Case Script

Step 1 On the **Testing Case > API Automation** page, click the name of the test case to be edited.

Step 2 Click the **Scripting** tab.

Step 3 Click .

Step 4 In the displayed dialog box, click the authentication mode drop-down list.

Step 5 Select one of the following modes and configure it by referring to the following information:

- **No Auth:** No authentication information is required.
- **Bearer Token:** Enter the authentication information in the token text box. Or use built-in functions, environment parameters, and local parameters. Up to 20,000 characters are supported.
- **API Key:** Enter a name and a value. Both support up to 20,000 characters. The name can contain only letters, digits, underscores (_), and hyphens (-). The value can use built-in functions, environment parameters, and local parameters.

Step 6 Click **OK**.

----End

Setting Authentication in a URL Request

Step 1 On the **Testing Case > API Automation** page, click the name of the test case to be edited.

Step 2 Click the **Scripting** tab.

Step 3 On the bottom of the page, click **URL Request** to generate a test step.

Step 4 Click the **Authentication** tab.

Step 5 Select one of the following authentication modes from the drop-down list box:

- **Inherit:** (Default) Inherits the authentication settings of the API automation test case script set in [Setting Authentication in an API Automation Test Case Script](#).
- **No Auth:** No authentication information is required.
- **Bearer Token:** Enter the authentication information in the token text box. Or use built-in functions, environment parameters, and local parameters. Up to 20,000 characters are supported.
- **API Key:** Enter a name and a value. Both support up to 20,000 characters. The name can contain only letters, digits, underscores (_), and hyphens (-). The value can use built-in functions, environment parameters, and local parameters.

Step 6 Click  on the top of the test step to save the configuration.

----End

6.4.4 Adding an API Script by Importing a Postman File

Background

Test steps can be generated by importing Postman files for automated API test cases.

Requirements:

- The Postman Collection v2.1 standard is supported.
- Use only the Postman request method, URL, header, and body to generate test steps.
- Use only the form-data, x-www-form-urlencoded, or raw modes to import the Postman request body.
- Upload the Postman request form-data attachment separately in test steps.

Procedure

Step 1 After the operations described in [Creating an Automated API Test Case Template](#) are complete, click the test case name, and click the **Scripting** tab.

Step 2 Click **Postman**. In the displayed dialog box, click , select the file to be imported, and click **OK**.

Step 3 The URL request is displayed on the page. Click **Save** in the upper right corner. The system displays a message indicating that the update is successful.

If the saving fails, a dialog box is displayed in the upper right corner, indicating the failure cause.

----End

6.4.5 Adding an API Script by cURL

Background

cURL is a command-line tool for transferring data specified with URL syntax. API automation allows you to copy the cURL of an API request from the Chrome control panel and paste the cURL to automatically generate an API test script.

Procedure

- Step 1** After the operations described in [Creating an Automated API Test Case Template](#) are complete, click the test case name, and click the **Scripting** tab.
- Step 2** Use a new browser tab to open the web page of the API to be tested and press **F12** to open the control panel.
- Step 3** Click the **Network** tab, right-click the API request in the list, and choose **Copy > Copy as cURL (bash)** from the shortcut menu.
- Step 4** Return to the **Scripting** tab, right-click the blank area on the page, and choose **Add Control > cURL** from the shortcut menu.

The test script is automatically generated on the page.

-----End

6.4.6 Adding an API Script by Keyword Library

6.4.6.1 Introduction to Keyword Library

Background

Keyword-driven testing is a test automation technique that creates automated test cases by providing a set of "build blocks" called keywords. Keyword-driven testing can be used at different test levels, such as component testing and system testing. Its advantages lie in usability, understandability, maintainability, reuse of test information, support for test automation, saving potential costs, and promoting progress.

When designing test cases, you may often use the same pre-steps or test logic. If these steps are written in each test case, the workload is heavy and the maintenance is difficult. Test keywords can help reuse these test steps.

Keyword Library manages API keywords, system keywords, and combined keywords in a unified manner to build one-stop keyword management capabilities and ensure consistent user experience during test case scripting.

- API keywords define the request of a single API, and can be generated by importing a Swagger file or saving a user-defined URL request.

For details about API keywords, see [Saving Test Procedure as an API Keyword](#).

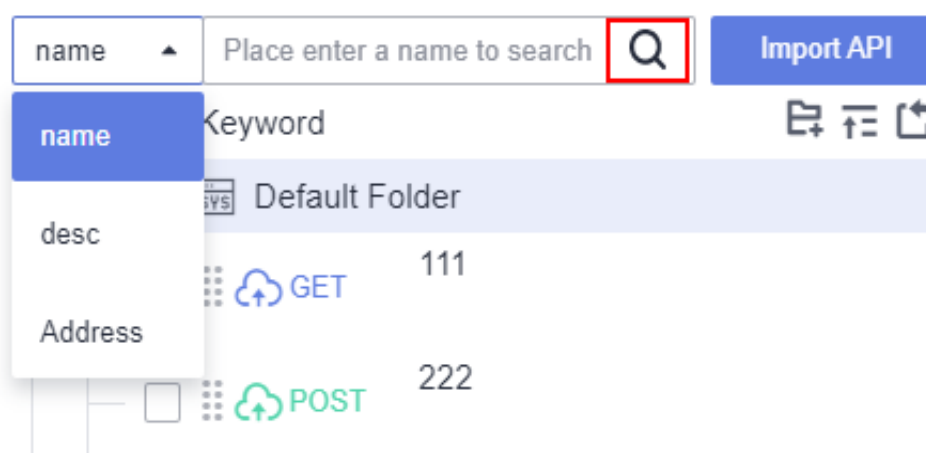
- Combined keywords encapsulate multiple test steps as common test logic. This test logic can be reused when the combined keywords are invoked by other test cases. For details about combined keywords, see [Saving Test Procedure as a Combined Keyword](#).
- System keywords cover authentication, protocols, middleware, and database, for various scenarios such as identity authentication, complex protocols, data processing, data preconfiguration, data verification, and API integration. For details about system keywords, see [System Keywords](#).

Accessing Keyword Library

- Step 1** After the operations described in [Creating an Automated API Test Case Template](#) are complete, choose **Testing Case** > **Auto API Test** and click the name of the test case to be edited.
- Step 2** Click the **Scripting** tab. **Keyword Library** is displayed on the right of the page. (If it is not displayed, click )
- The Swagger import keywords are placed on the **API** tab page. For details, see [Saving Test Procedure as an API Keyword](#).
 - Keyword test cases and combined keywords are placed on the **Combine** tab page. You can create combined keywords from test case keywords and then save them. For details, see [Saving Test Procedure as a Combined Keyword](#).
 - Practical keyword types such as authentication, database operation, middleware, and protocol are placed on the **System** tab page. For details, see [automated API test keywords](#).
- Step 3** Click the drop-down list on the **API** tab page and search with keywords of name, description, or address.

Action Word Lib

API Combine



----End

6.4.6.2 Saving Test Procedure as an API Keyword

Background

API keywords define the request of a single API and can be generated by importing a Swagger file or saving a user-defined URL request.

Swagger is a tool for defining, developing, and debugging APIs such as RESTful. Swagger can be used to define API attributes in a standardized manner, facilitating interconnection and interworking. API automation allows you to import API description files in Swagger 2.0 and 3.0 format, parse API definition descriptions, and generate a script template. You only need to enter API parameters based on the template to create automated API test cases.

You can import a Swagger API description file to generate a script template. A script template corresponds to an API definition in the Swagger file. Test cases can be orchestrated in a visualized manner based on the script template.

The following tables show the mapping between the script template and the fields in the Swagger API description.

- Swagger 2.0 format

Script Template Attribute	Swagger API Definition Attribute
Name	By default, operationId is used. You can set it to summary .
Path	schema + :// + basePath + path.
Request parameter hostURL	host
Other request parameters	parameters

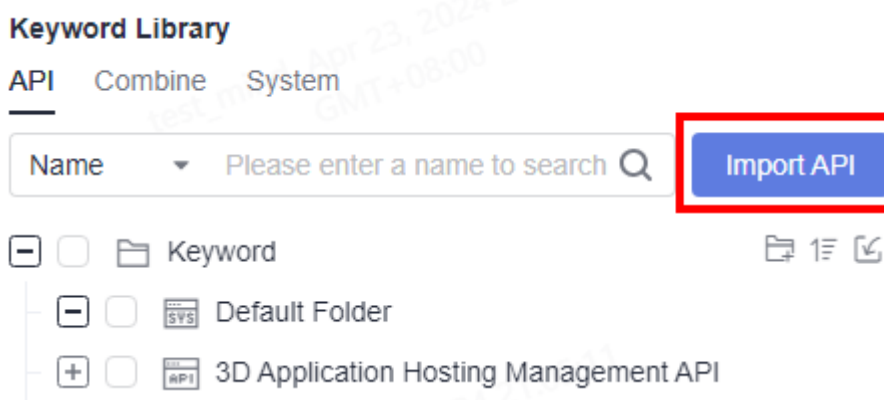
- Swagger 3.0 format

Script Template Attribute	Swagger API Definition Attribute
Name	By default, operationId is used. You can set it to summary .
Path	url + path
Request parameter hostURL	servers: - URL: https://{hostURL}/variable variables: hostURL: default: test.demo.com

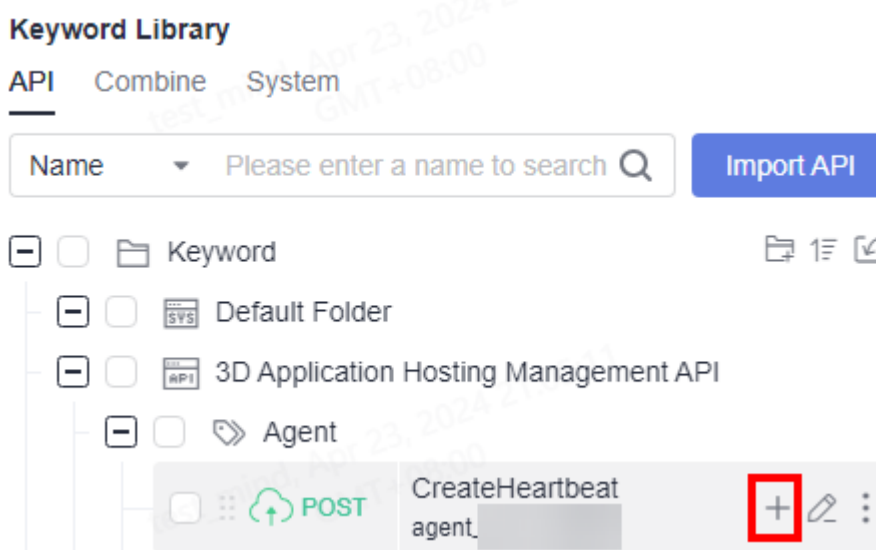
Script Template Attribute	Swagger API Definition Attribute
Other request parameters	parameters , requestBody , and responses

Importing a Swagger File to Generate a Test Script

- Step 1** After the operations described in [Creating an Automated API Test Case Template](#) are complete, click the test case name, and click the **Scripting** tab.
- Step 2** On the **Keyword Library** page on the right, click **Import API**.



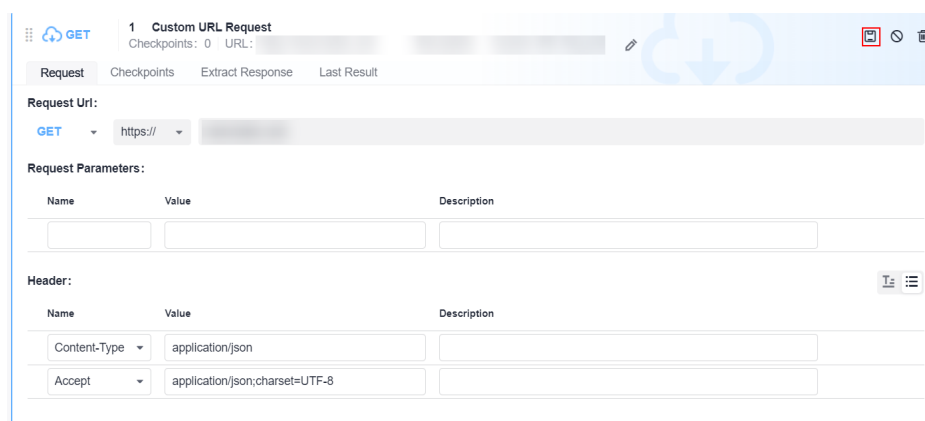
- Step 3** Click **Click to add file or drag upload**, select the configured Swagger API file, and click **OK**.
- Step 4** After the file is imported successfully, the system automatically parses and generates a script template. The script template contains the basic description of the API. You can click **+** or drag and drop an API on the **Keyword Library** page to add it to the test step. You only need to enter the API parameters based on the template to perform the test.



----End

Saving a Customized API Keyword

- Step 1** After the operations described in [Creating an Automated API Test Case Template](#) are complete, click the automated API test case name and click the **Scripting** tab.
- Step 2** After the operations described in [Adding an API Test Script by Using a Custom URL Request](#) are complete, select the test step to be set as a keyword (only for custom URL request steps), and click  at the upper right corner of the test step page to save the API keyword.



- Step 3** On the displayed page, set **API Keyword** and **Description**, and select the directory for storing the keyword. By default, keywords are stored in **Default Folder** under **API > Keyword**.

Create Custom API Keyword ✕

* API Keyword :

Enter a keyword.

Description:

Custom URL Request

* Storage Folder:

[-] Keyword

Default Folder

Save

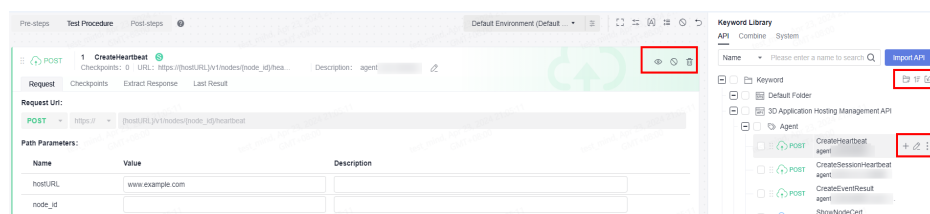
Cancel

Step 4 Click **Save**.

----End

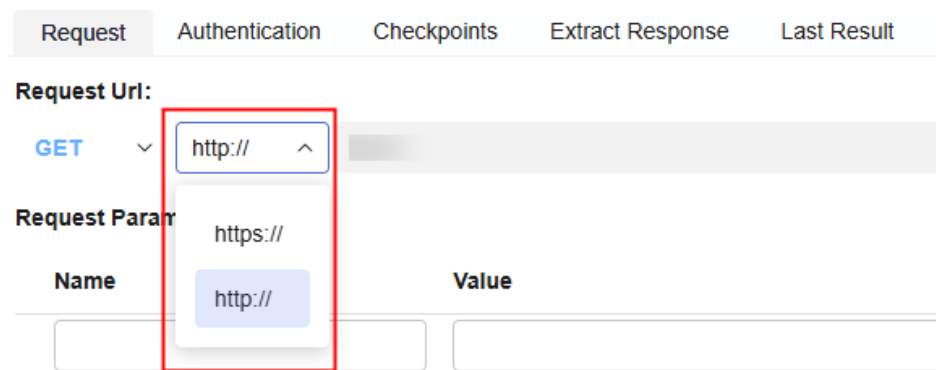
Managing Existing Keywords

For an existing keyword, you can hover the cursor over the keyword area and perform the following operations:



- Click a keyword name to view the details about the keyword.
- Click  on the right of the **Keyword** to create a folder. Save the keyword set in [Step 3](#) to a custom folder.

- Hover the cursor over the left area of a keyword name to adjust the keyword sequence.
- Click  or hover the cursor over the keyword and drag it to the blank page of the test step to add a test step as a keyword.
- Click any area of a keyword to edit it.
- Click  to view the information about the key word.
- Click  to disable a keyword, and click it again to enable the keyword.
- Click  to delete the keyword from the test procedure.
- Modify the protocol type of a saved keyword.



Name	Value
	http://

6.4.6.3 Saving Test Procedure as a Combined Keyword

Background

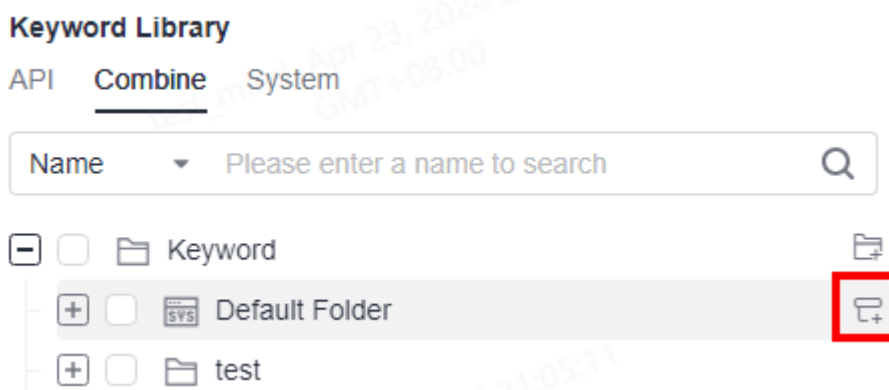
When designing test cases, you may often use the same pre-steps or test logic. If these steps are written in each test case, the workload is heavy and the maintenance is difficult. Combined keywords encapsulate multiple test steps as common test logic. This test logic can be reused when the combined keywords are invoked by other test cases.

Scenario 1

- Step 1** Click  in the upper right corner of the **Scripting** tab page.
- Step 2** Enter the **Name** and **Description**, select the directory, and set request parameters as required. Select existing URL requests or request headers and click **Save**.
- Step 3** In the **Keyword Library > Combine** tab page, view the combined keywords that have been saved.
- End

Scenario 2

- Step 1** On the **Keyword Library > Combine** page, click  next to the folder to be saved.



Step 2 Set **Name** and **Description**.

Step 3 (Optional) Configure request parameters and authentication in the combined keyword.

Step 4 Add custom URL requests (refer to [Adding an API Test Script by Using a Custom URL Request](#)) and common request headers to the combined keyword.

NOTE

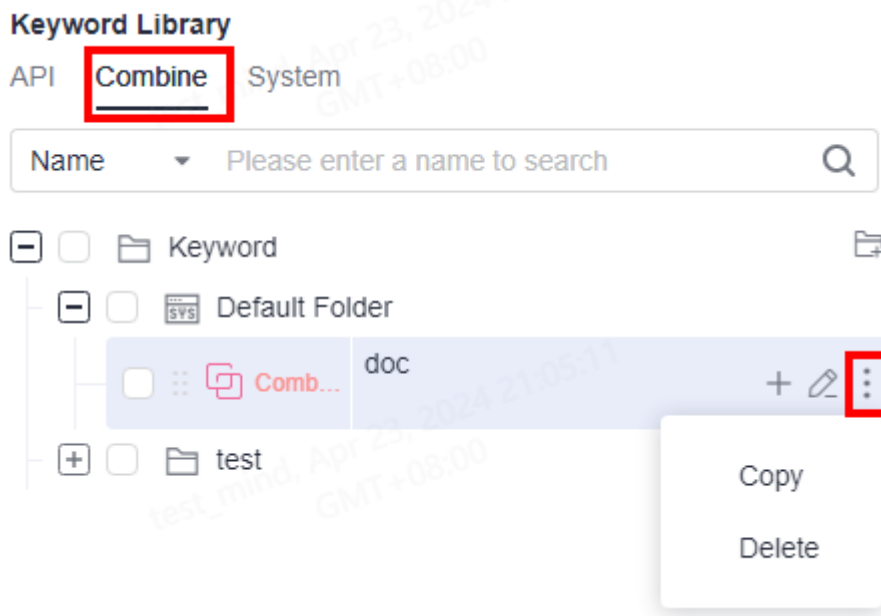
- To invoke the request parameter that has been set in [3](#), use **\$** in the **Value** column.
- To inherit the authentication mode that has been set for the whole combined keyword ([3](#)), click the **Authentication** tab of a request and select **Inherit** under **Authentication Mode**.

Step 5 Click the **API** tab, select the folder where the keyword to be added is located, and click on the right of the keyword to be added or drag it to the test step area.

Step 6 Click **Save**.

----End

For an added combined keyword, you can hover the cursor over the keyword area and perform the following operations:



- Click  on the right of the **Keyword** to create a folder. You can create combined keywords in a custom folder by referring to [Step 1](#).
- Click  or hover the cursor over **Combine** to drag the combined keyword to the blank page of the test step to add it to the script.
- Hover the cursor over **Combine** to adjust the sequence of combined keywords.
- Click  to edit the keyword information.
- Click **Mark As**. You can set the status of **Normal**, **New**, or **Update** for the combined keyword.
- Click **Copy** to copy a combined keyword test case in the folder.
- Click **Delete** to delete a combined keyword.

Modifying the Protocol and Address of a Combined Keyword

Modify the protocol and address of a combined keyword or URL request.

Step 1 On the **Keyword Library** > **Combine** tab page, find the target combined keyword.

Step 2 Hover the cursor over the combined keyword name and click .

Step 3 Modify the saved protocol type.

Step 4 Click **Save**.

----End

6.4.6.4 Refreshing a Keyword in Multiple Test Cases

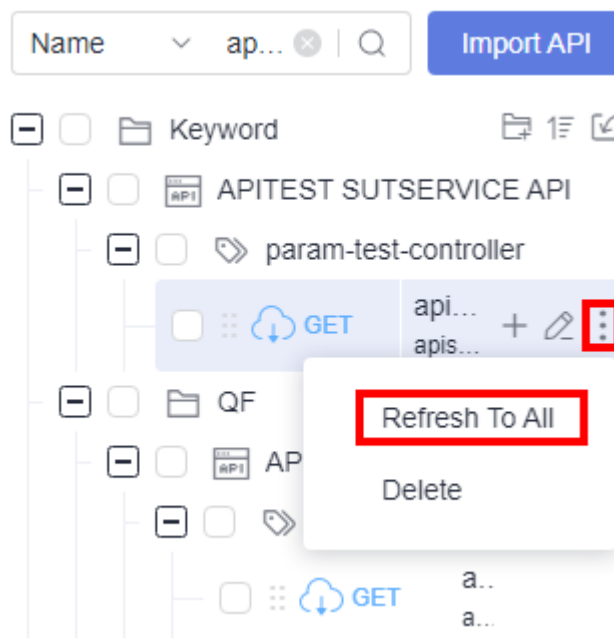
If you have used a keyword to write test cases and then want to modify the keyword in all these test cases, you can refresh the keyword. At present, this refresh feature is limited to combined keywords and the keywords generated by importing YAML files.

Prerequisites

You have used a keyword in test cases or combined keywords.

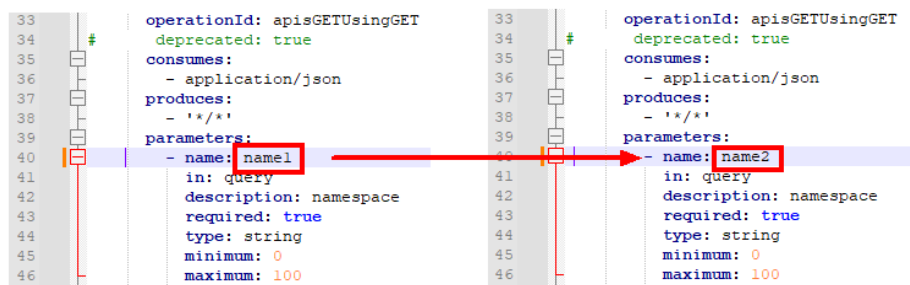
Procedure

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** On the **Auto API Test** tab page, click a case name and click the **Scripting** tab.
- Step 4** On the **Keyword Library** part on the right, click  next to the keyword and click **Refresh to All**.

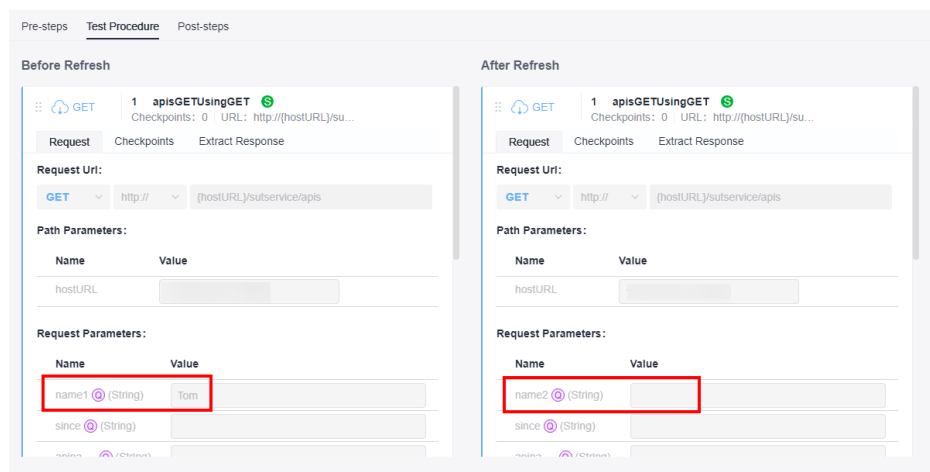


- Step 5** On the right of the displayed dialog box, configure the refresh rule. Parameter values can be inherited. Example:

Change the parameter name **name1** in the YAML file to **name2**.



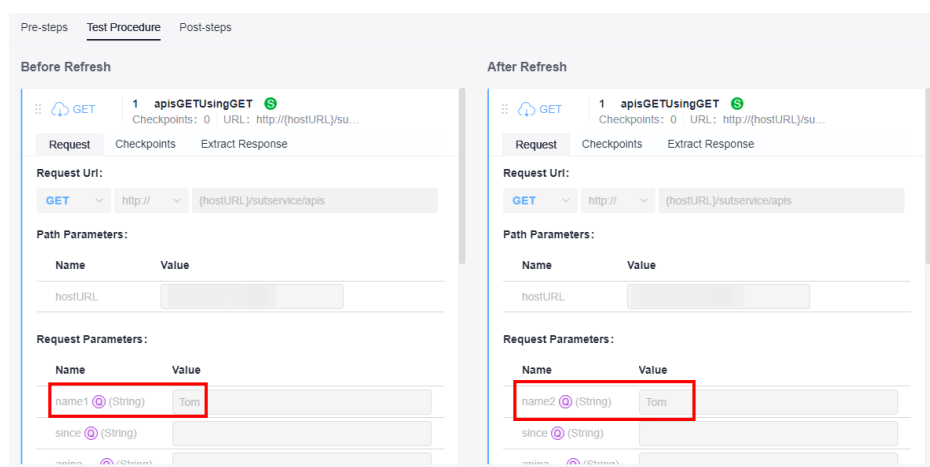
Import the new YAML file. The value of **name1** is not automatically inherited to **name2** yet.



Configure a refresh rule. The figure below shows an example.



Click **OK**. The value is inherited.



NOTE

For the refresh rule:

- Parameter name must be unique.
- The content in the text box of script will not refresh with the keyword.

Step 6 On the left of the dialog box for refreshing keyword, you can configure whether to refresh the keyword in the test cases that were removed to the recycle bin.

- Test cases marked with  are in the recycle bin.
- Test cases marked with  can be used normally.

Step 7 If there are cases or combined keywords that you do not want to refresh, click next to them to remove them from the list. These removed items can still be searched.

Step 8 Click **OK**.

CAUTION

Refresh operation cannot be rolled back. Exercise caution before performing this operation.

----End

6.4.7 Adding Logic Control to an API Script

Logic control is used to orchestrate test scenarios, including sleep, group, if condition, and loop.

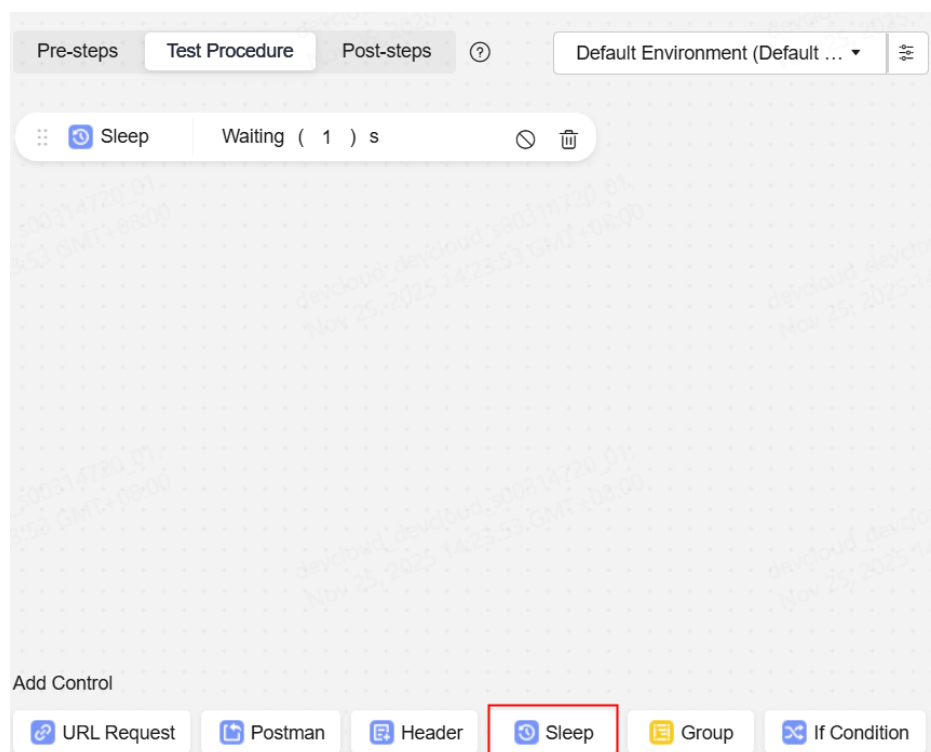
Sleep

Sleep allows you to wait for a period of time after performing a test step.

To set **Sleep**, perform the following steps:

Step 1 Go to the **Scripting** tab page and click **Sleep**.

Step 2 Enter an integer from 1 to 60.



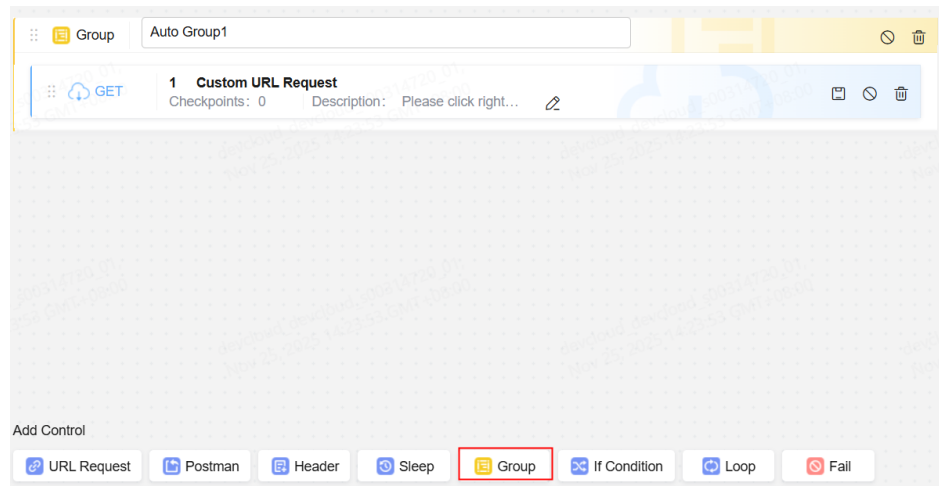
----End

Group

Step 1 Go to the **Scripting** tab page and click **Group**.

Step 2 Enter the group name and drag related test steps to the group.

- The **URL Request**, **If Condition**, **Sleep**, and **Loop** test steps can be grouped.
- You can drag and orchestrate the sequence of groups in test cases.
- You can drag and orchestrate the sequence of test steps in a group.
- You can disable or delete all groups.



----End

if Condition

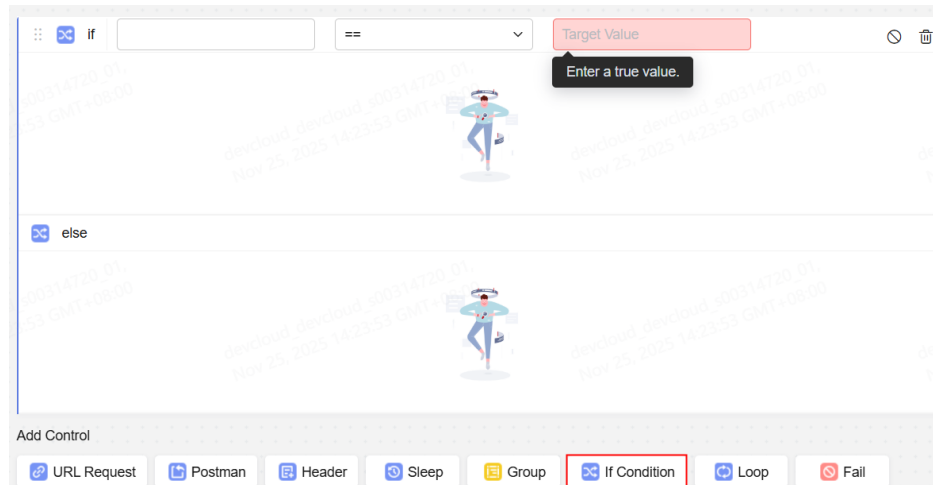
To determine the subsequent test steps based on the results of the previous test steps, use **If Condition**.

To set an **if** condition, perform the following steps:

Step 1 Go to the **Scripting** tab page and click **If Condition**.

Step 2 Enter the parameter, and add subsequent test steps to the branch.

URL Request, **If Condition**, **Sleep**, and **Loop** can be added to the branch of the **If Condition**.



----End

NOTE

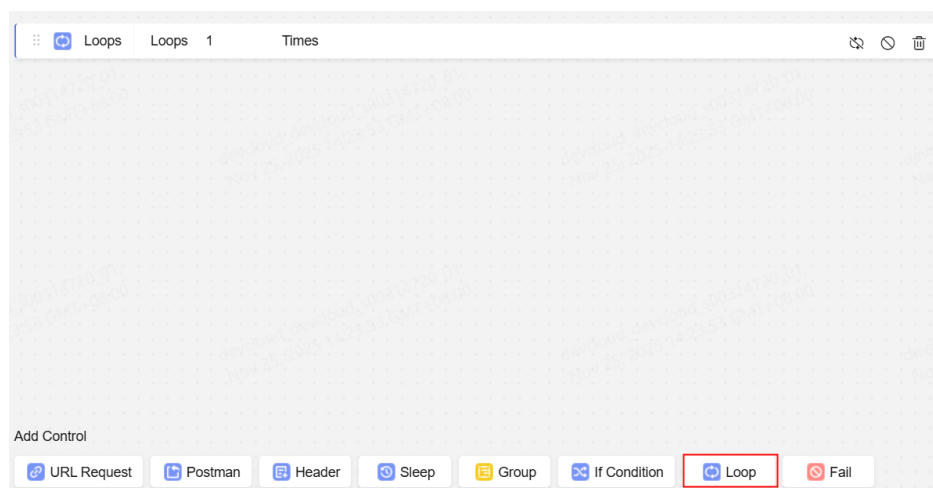
The **if** condition supports referencing local variables and passing parameters. For details about how to use comparison operators, see [Comparison Operator Description](#).

Loop

Step 1 Go to the **Scripting** tab page and click **Loop**.

Step 2 Set the loop count.

You can add URL requests, if conditions, groups, wait time, and test keywords to a loop.



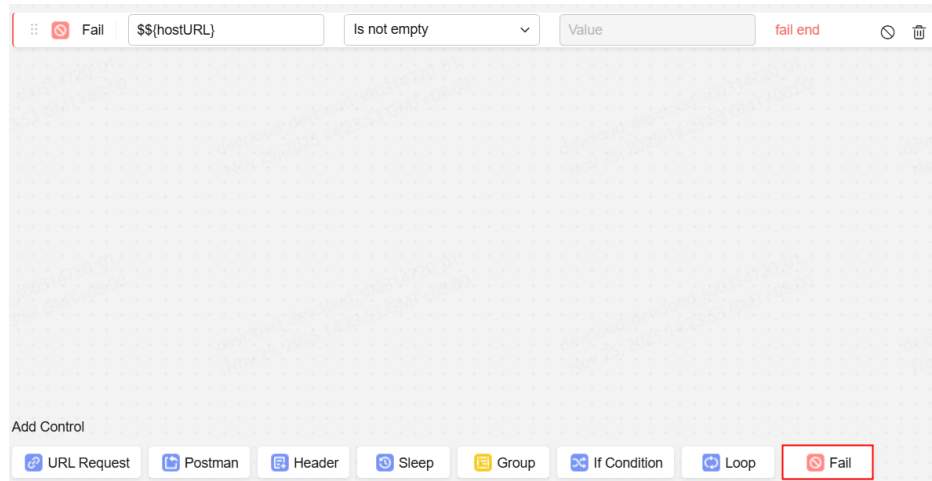
----End

Termination

To terminate the polling when certain condition is not met during the execution, add a **Fail** control step.

Step 1 On the **Scripting** tab page, click **Fail**.

Step 2 Set the condition parameter. (Only environment and local parameters are supported.) If the condition is met, the test case fails.



----End

6.4.8 Setting Test Case Parameters of an API Script

Background

Proper test design requires that the test logic and test data be separated to maximize the reuse of the test logic and enhance the maintainability and input-to-output ratio of test cases. For example, the URL domain names of different test environments are test data independent of the test logic and related to the test environment. Test case parameters can be used to manage test data.

Parameters of automated API test cases are classified into the following types:

- **Local Parameters**
- **Built-in Parameters**
- **Response Extraction Parameters**

Masking Sensitive Parameters

Sensitive parameters involve sensitive information and security issues, such as typical sensitive data (passwords and bank accounts), common authentication credentials (keys and private keys), and personal data. These parameters must be protected. You can mask sensitive information in the API request response parameters. For details, see [Step 7](#).

Local Parameters

Local parameters can be used in the current test case. For example, parameters, checkpoints, and variables of test steps can reference local parameters.

Use `$${Parameter name}` to reference a local parameter, for example, `$${hostIp}`.

The following tables show the main configuration items of local parameters.

Table 6-2 Main local parameter configuration items

Configuration Item	Mandatory	Description
Name	Yes	The value can contain 1 to 300 characters. Only letters, digits, periods (.), underscores (_), and hyphens (-) are supported.
Type	Yes	Supported types: text, random character string, random integer, timestamp, formatted timestamp, UUID, Base64 encoding, MD5 hash value, password or authentication information, and SHA-512 encoding. For details, see Table 6-3 .
Description	No	Brief description of the parameter (current parameter by default). The value can contain up to 3,000 characters. Click the text box to enter the description. Click  , enter a JSON string, click JSON Parse to add line breaks and indentation, and click Fill Back .
Value	No	Local parameter values. Assign values to the supported parameter types by referring to Table 6-3 .
Sensitive	No	CodeArts TestPlan encrypts sensitive parameter values during storage and overwrites them with asterisks (*) in test result logs. Sensitive data includes but is not limited to personal and authentication information, such as names, addresses, and usernames.
Dynamic	No	Values of dynamic parameters can be assigned during test case execution. The initial value of a dynamic parameter can be empty. After a value is assigned, the latest value is displayed. Assigning values to dynamic parameters: In the Extract Response tab page of a test step, set the value of the Assign Value to Dynamic Environment Parameter column. The extracted value will be assigned to the dynamic parameter during test execution.

Table 6-3 Parameter types

Name	Description
Text	The value can contain up to 10,000 characters. A local parameter of text type can be set to Sensitive and Dynamic , which are not selected by default.

Name	Description
Random character string	The system randomly generates a character string of a length from 1 to 9999 digits. The parameter of this type cannot be set to Sensitive or Dynamic .
Random integer	The system randomly generates an integer from -999999999 to 999999999. The parameter of this type cannot be set to Sensitive or Dynamic. For example, if this parameter is set to [-9999,9999], a random integer in this range will be generated.
Timestamp	The system generates the current integer timestamp. No value needs to be set. The parameter of this type cannot be set to Sensitive or Dynamic .
Formatted timestamp	The default value is the current timestamp. Click the timestamp format and select the required format from the drop-down list. For details about the formats, see Table 6-5. The parameter of this type cannot be set to Sensitive or Dynamic. Example 1: For yyyy-MM-dd HH:mm:ss:33250825252000, the expected value is 3023-09-05 20:20:52. Example 2: For yyyy-MM-dd:33250825252000, the expected value is 3023-09-05.
UUID	No value needs to be set. The parameter of this type cannot be set to Sensitive or Dynamic .
Base64 encoding	The parameter of this type uses Base64 encoding method. The value can contain up to 256 characters. The parameter of this type cannot be set to Sensitive or Dynamic .
MD5 hash value	The system generates an MD5 hash value with the parameters of this type. The value can contain up to 256 characters. The parameter of this type cannot be set to Sensitive or Dynamic .
Password or authentication information	The value can contain up to 256 characters. The parameter of this type is set to Sensitive by default and cannot be changed.
SHA-512 encoding	The value can contain up to 256 characters. The parameter of this type is set to Sensitive by default and cannot be changed.
Array	The value is in JSON format and contains up to 10,000 characters. This parameter cannot be set as sensitive or dynamic.

To set local parameters, perform the following steps:

- **Creating a Variable**

Step 1 Go to the **Scripting** tab page and click .

Step 2 Click **New Variable**. Enter the parameter name, type, and value.

After all the parameters are set, click **Save**.

----End

- **Importing Variables in Batches**

Step 1 Go to the **Scripting** tab page and click .

Step 2 Click **Import**.

Step 3 In the displayed dialog box, click **Download Template**.

Set parameters based on the format requirements in the template and return to the page. Click , upload the file, and click **Confirm**.

Step 4 View the import result.

- Import successful: New parameters are displayed in the list. The number of new parameters is the same as the number of rows in the Excel file.
- Import failed: A dialog box is displayed in the upper right corner of the page, indicating that the import fails. Click **Details** to view the error cause, rectify the fault, and import the file again.

----End

Built-in Parameters

Built-in parameters are used to parameterize the corresponding part of an HTTP or HTTPS response. You can select built-in parameters from the **Source** options in the **Checkpoints** and **Response** pages.

The following table lists the built-in parameters of CodeArts TestPlan.

Built-in Parameter	Description	Multi-level Values Supported or Not	Function	Example
Response Body (JSON)	Response body returned by an API.	Yes	- Checkpoint property field. - Property field for parameter passing.	- Checkpoint: Check whether the value of id in the response body is 100. - Settings: Set Source to Response body (JSON), Property to id (only if the id field exists in the JSON string), Comparison Operator to Equals (case insensitive), and Target Value to 100.
Response Header	Response header returned by an API.	Yes	- Checkpoint property field. - Property field for parameter passing.	- Checkpoint: Check whether the value of token in the response header is abcd. - Settings: Set Source to Response Header, Property to token (only if token exists in the response header), Comparison Operator to Equals (case insensitive), and Target Value to abcd.
Response Code	Response code returned by an API.	No	- Property or value of a checkpoint. - Property field of a variable.	- Checklist: Check whether the response code is 200. - Settings: Set Source to Response Code, Comparison Operator to Equals, and Target Value to 200.

 NOTE

Built-in parameters support multi-level values, for example:

- If the response body is `{"result":{"appId":12}}`, the format of **appId** is as follows:
Source=Response body; Property=result.appId. If the value of **result** is in array format, **Property=result[i].appId**, where *i* is a non-negative integer.

Response Extraction Parameters

Response extraction parameters are extracted from the response body of an API. For details about the definition and usage, see [Setting the Response Extraction of an API Script](#).

6.4.9 Setting Environment Parameters of an API Script

During automated testing, there are multiple test environments whose parameter values are different, for example, the domain name and account. These parameters are often used in test scripts. If they are bound to test scripts, the scripts will become highly redundant and hard to be reused.

To solve the preceding problems, you can use environment parameters to manage them in a unified manner. In test scripts, environment parameters are referenced in parametrization mode. During execution, you only need to select an execution environment to use the corresponding environment parameter values to complete testing.

Application Scope

The parameters, checkpoints, variables, and URLs of the test steps in each test case of the current project can reference environment parameters.

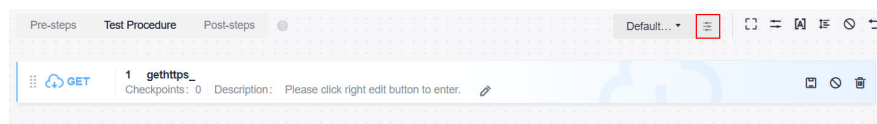
Reference Format

The reference format of an environment parameter is **`\${Parameter name}`**. For example, if the parameter name is **hostname**, you can use **`\${hostname}`** to reference the parameter.

You can configure and manage environment parameters by group as required. For example, the value of **hostname** in the quasi-production environment is **stage.example.com**, and the value in the production environment is **prod.example.com**. The test script uses **`\${hostname}`** to reference this parameter. During the test, select different environments to execute the test so that a set of automated API test cases can be reused in all environments.

Procedure

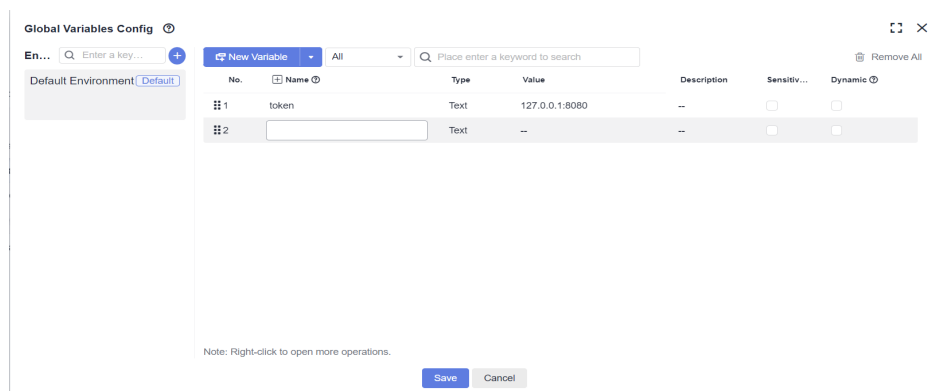
Step 1 Go to the **Scripting** tab page and click .



Step 2 Click **New Variable**, enter parameter information, and click **OK**.

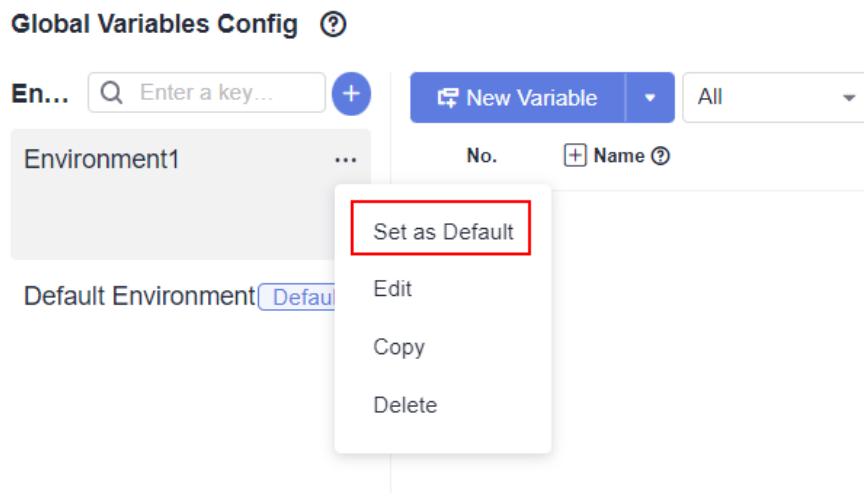
Configuration Item	Description
Name	Parameter name.

Configuration Item	Description
Type	Parameter type. Multiple types are supported, such as text, random string, random integer, and timestamp.
Value	Parameter value.
Description	Parameter description.
Sensitive	This service encrypts sensitive parameter values during storage and overwrites them with asterisks (*) in test result logs. Sensitive data includes but is not limited to personal and authentication information, such as names, addresses, and usernames.
Dynamic	Dynamic parameter setting. The value of a dynamic parameter can be assigned during test case execution. The initial value of a dynamic parameter can be empty. After a value is assigned, the latest value is displayed. After you set the value of the Assign Value to Dynamic Environment Parameter column in the Extract Response tab page of the test step, the extracted value is assigned to the dynamic parameter during test execution. For details, see Dynamic Variable Description .



Step 3 Change the default environment.

To set another environment as the default environment, click **...** in the upper right corner of the environment card and select **Set as Default**.



-----End

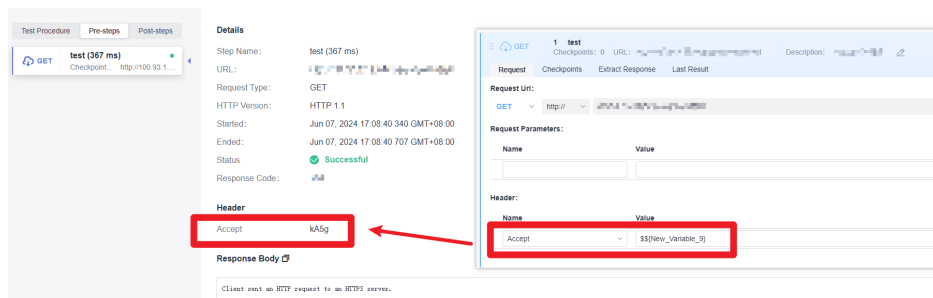
Built-in Functions

1. You can use built-in functions to configure environment parameters. Note that nested built-in functions are output as they are.
2. For details about the supported built-in functions, see [Table 6-4](#).

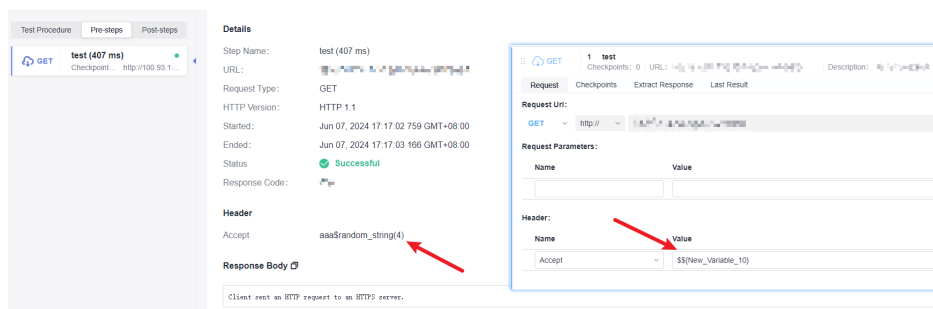
Table 6-4 Supported built-in functions

No.	Function	Description
1	<code>\$random_string(intA)</code>	Generate a random string of a specified length
2	<code>\$random_int(intA, intB)</code>	Generate a random number in a specified range
3	<code>\$timestamp()</code>	Obtain the current timestamp
4	<code>\$timestamp_format(String A, String B)</code>	Convert a timestamp into a date
5	<code>\$uuid()</code>	Generate a UUID
6	<code>\$encode_base64(StringA)</code>	Generate Base64 encoding
7	<code>\$md5(StringA)</code>	Generate an MD5 hash value
8	<code>\$sha512(StringA)</code>	Generate SHA512 encoding

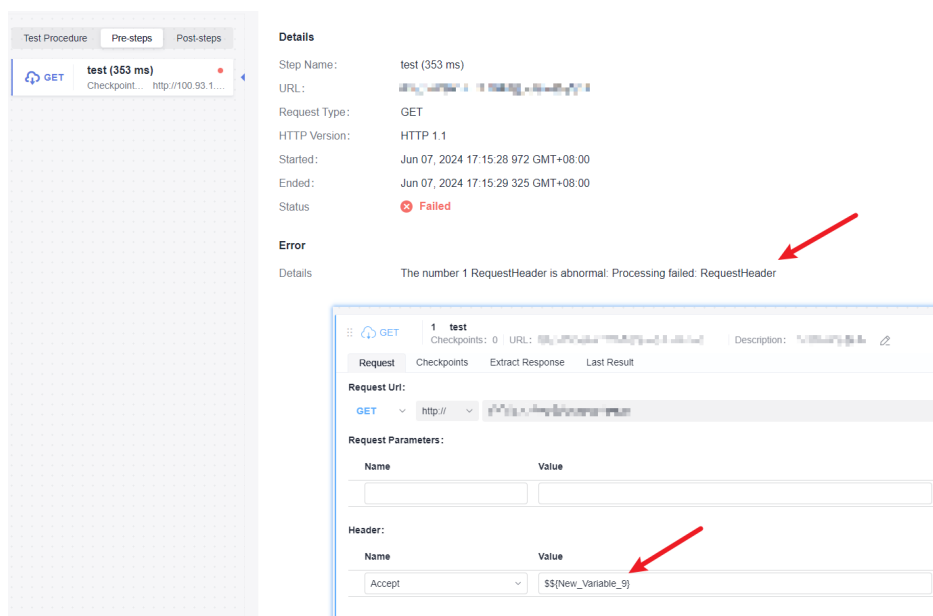
3. Examples:
 - The value of `$random_string(intA)` is output normally.



- Built-in functions that concatenate strings are parsed as strings, and their output values are strings as they are. For example, the output values of functions `aa$random_string(10)` and `$random_string(10)aa` are `aa$random_string(10)` and `$random_string(10)aa`.

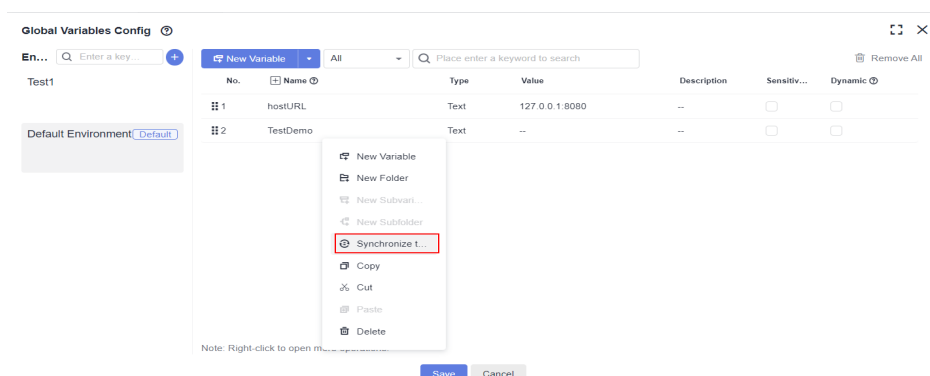


- If two parameters are passed while only one parameter is required, for example, `$random_string(1,1)`, an exception will be thrown.
If the input and required parameter types do not match, for example, `$random_string(fff)` or `$random_string()`, an exception will be thrown.



Synchronization

Right-click the parameter to be synchronized and choose **Synchronize to other environments** from the shortcut menu to synchronize the current parameter to all environments.



Copy Environment Parameters in an Environment or to Other Environments

Step 1 Right-click the parameter to be copied and choose **Copy** from the shortcut menu.

Step 2 Go to the environment page list of the current environment or other environments, right-click the blank area, and choose **Paste** from the shortcut menu.

-----End

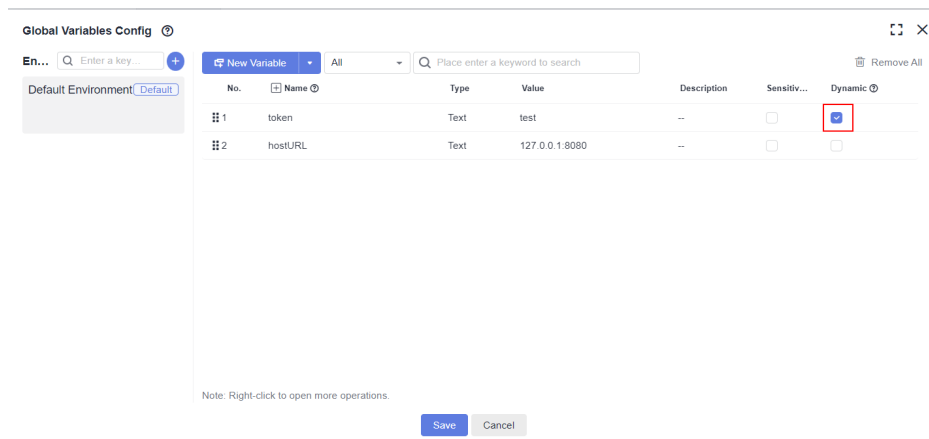
Dynamic Variable Description

Multiple test cases in a test suite may have context relationships. The test case that is executed later depends on the result returned by the API in the test case that is executed earlier. For example, all APIs require authentication information, which has a validity period. If it is obtained in each test case, the test procedure of the test case will be redundant and difficult to maintain.

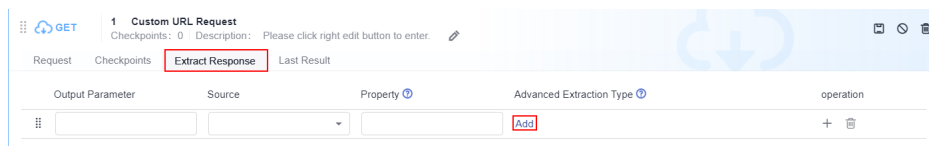
This problem can be avoided by using dynamic global variables. In the first executed test case, the authentication information is obtained and then assigned to the dynamic global variables. In the subsequent executed test cases, the dynamic global variables can be directly used, avoiding obtaining authentication information repeatedly.

Step 1 Set dynamic variables.

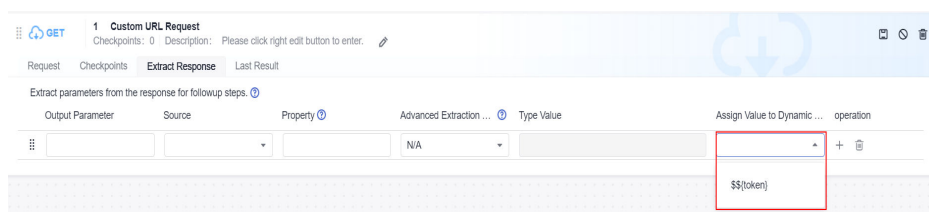
1. On the test case list of the **Auto API Test** tab page, click **More** on the right of the page and choose **Environment Parameters**.
2. Select the check box in the **Dynamic** column and click **Save** to set a global variable to a dynamic variable.

**Step 2** Assign values to dynamic variables.

1. In an automated API test case, click the **Extract Response** tab of the test step, and click the **Add** button in the **Advanced Extraction Type** column.

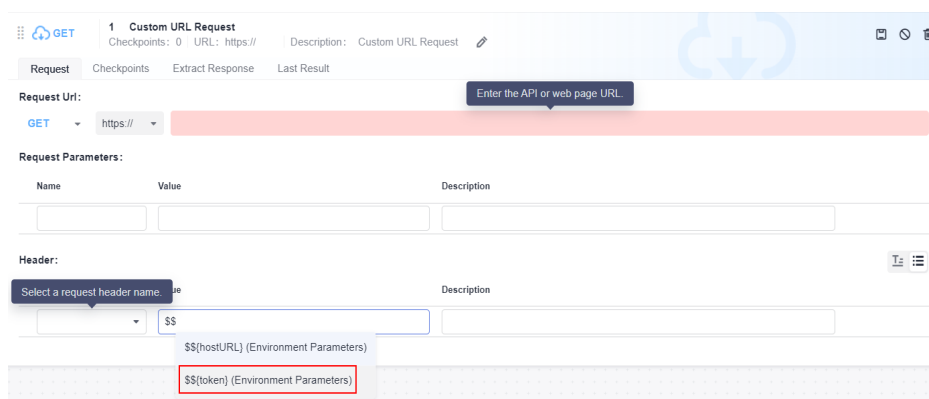


2. Select a value from the **Assign Value to Dynamic Environment Parameter** drop-down list box.

**Step 3** Use a dynamic variable.

- Use the dynamic variable in a test case.

Reference the dynamic global variable in a test case as required. For details, see [Reference Format](#).



- Use a dynamic global variable in a test suite.

Add the test cases in [Step 2](#) and [Step 3](#) to a test suite in sequence, and select **Serial**. In this way, the latest value assigned to the dynamic global variable can be used during the execution of related test cases.

 NOTE

You are not advised to use dynamic global variables during parallel execution because the value assigning and sequence of the dynamic variables are uncertain.

----End

6.4.10 Importing an Automated API Test Case Dataset

API testing may involve testing a logic in multiple rounds with different data combinations. Manually configuring test case scripts for each round is labor- and time-consuming.

With CodeArts TestPlan, you can perform data-driven API testing by importing Excel files of test data to efficiently generate and execute automated API test cases for different test scenarios.

 NOTE

Currently, this feature is available in the **AP-Singapore** and **LA-Mexico City** regions.

Prerequisites

1. You have configured an accessible request address in the script of the target API test case.
2. You have set the fields in the Excel dataset file as [local parameters](#).

Constraints

1. A maximum of 100 rows and 50 parameter fields can be uploaded.
2. Only one Excel file can be uploaded each time.
3. By default, API automation test case datasets are executed using the **DEFAULT** agent pool that is set for the API cases. They cannot be executed by custom executors.
4. If the names and sensitivity of local parameters related to an uploaded dataset change, upload the dataset file again to ensure correct display and execution.

Procedure

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Click the **Auto API Test** tab, click the target test case, and click the **Scripting** tab.

Step 3 Click .

Step 4 In the displayed dialog box, click the **Default Environment** drop-down box and select the target environment. For details about how to set the environment, see [Setting Environment Parameters of an API Script](#).

Step 5 Click **Download Template** and save the Excel template file to the local PC.

Step 6 Fill in the dataset description and local parameters. If multiple rows of data are set, the test case will be executed using each row round by round.

- **Dataset Description:** Optional. 1 to 128 bytes.
- Other headers must have been set as local parameters (case-sensitive) in the case scripting page. For details, see [Table 6-2](#).

Step 7 In the displayed dialog box, click , select the Excel file, and click **Upload**.

 NOTE

1. A maximum of 100 rows and 50 parameter fields can be uploaded.
2. Only one Excel file can be uploaded each time.
3. Uploading a new Excel file will clear the data of the previous file, and the dialog box will show the data from the most recent upload.

Step 8 The uploaded data is displayed in the list. To delete the data, click **Clear**.

Step 9 Click **Close**.

Step 10 Click a URL request, and use **\$** to reference the configured local parameters. When the test case is executed, the values of the local parameters are read. For details about the execution, see [Execution of an Automated API Test Case with a Dataset](#).

----End

6.4.11 Built-in Functions

6.4.11.1 Binary Addition Function

Function Name

`$add(intA, intB)`

Function Description

Implements the addition operation between parameter A and parameter B. Parameters A and B support the following types:

- Numbers
- Local parameters
- Binary operations

Application Scenarios

The binary addition function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body

- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of the **test** parameter in the request URL is the binary addition function. Parameters A and B in the function are **1000**.

Name	Value
test	\$add(1000,1000)

- Request header

As shown in the following figure, the value of parameter **add** in the request header is the binary addition function. Parameter A in the function is the binary subtraction function **\$subtract (1001,1000)**, and parameter B is **-1**.

Name	Value
test	\$strConcat(abc, 123)
add	\$add(\$subtract(1001,1000),-1)

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the binary addition function. Parameter A in the function is the local parameter **test**, and parameter B is **1**. For details about how to set local parameters, see [Local Parameters](#).

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		==	\$add(\$test,1)

- **if** condition

As shown in the following figure, the target value of the **if** condition is the binary addition function. Parameter A in the function is **1**, and parameter B is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

if	\$test	==	\$add(1,\$status)
----	--------	----	-------------------

- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the binary addition function. Parameters A and B in the function are **1000**.

Loops	Loops 10	Times	
Break Condition	\$(status)	>	\$add(1000,1000)

6.4.11.2 Binary Subtraction Function

Function Name

\$subtract(*intA*, *intB*)

Function Description

Implements the subtraction operation between parameter A and parameter B. Parameters A and B support the following types:

- Numbers
- Local parameters
- Binary operations

Application Scenarios

The binary subtraction function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of the **test** parameter in the request URL is the binary subtraction function. Parameters A and B in the function are **1001** and **1000** respectively.

Name	Value
test	\$subtract(1001,1000)

- Request header

As shown in the following figure, the value of parameter **subtract** in the request header is the binary subtraction function. Parameter A in the function is the binary multiplication function **\$multiply(100,100)**, and parameter B is **-1**.

Name	Value
Content-Type	application/json
subtract	\$subtract(\$multiply(100,100),-1)

- Request body

As shown in the following figure, the binary subtraction function is used in the request body. Parameter A in the function is the binary division function **\$divide(1000,100)**, and parameter B is the binary addition function **\$add(1000,1000)**.

```
1 {
2   "add": "$add($multiply(100,100),$divide(1000,100))",
3   "subtract": "$subtract($divide(1000,100),$add(1000,1000))",
4   "multiply": "$multiply($add(1000,1000),$subtract(1001,1000))",
5   "divide": "$divide($subtract(1001,1000),$multiply(100,100))"
6 }
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the binary subtraction function. Parameter A in the function is the local parameter **test**, and parameter B is **1**. For details about how to set local parameters, see [Local Parameters](#).

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		==	\$subtract(\$test, 1)

- if condition

As shown in the following figure, the target value of the **if** condition is the binary subtraction function. Parameter A in the function is **1001**, and parameter B is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

if	\$test	<	\$subtract(1001,\$status)
----	--------	---	---------------------------

- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the binary subtraction function. Parameters A and B in the function are **1001** and **1000** respectively.

Loops	Loops	10	Times
Break Condition	\$\$status	>	\$subtract(1001,1000)

6.4.11.3 Binary Multiplication Operation

Function Name

`$multiply(intA, intB)`

Function Description

Implements the multiplication operation between parameter A and parameter B. Parameters A and B support the following types:

- Numbers
- Local parameters

- Binary operations

Application Scenarios

The binary multiplication function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of the **test** parameter in the request URL is the binary multiplication function. Parameters A and B in the function are **100**.

Name	Value
test	\$multiply(100,100)

- Request header

As shown in the following figure, the value of parameter **add** in the request header is the binary multiplication function. Parameter A in the function is the binary division function **\$divide(1000,100)**, and parameter B is **-1**.

Name	Value
Content-Type	application/json
multiply	\$multiply(\$divide(1000,100),-1)

- Request body

As shown in the following figure, the binary multiplication function is used in the request body. Parameter A in the function is the binary addition function **\$add(1000,1000)**, and parameter B is the binary subtraction function **\$subtract(1001,1000)**.

```
1  {  
2      "add": "$add($multiply(100,100),$divide(1000,100))",  
3      "subtract": "$subtract($divide(1000,100),$add(1000,1000))",  
4      "multiply": "$multiply($add(1000,1000),$subtract(1001,1000))",  
5      "divide": "$divide($subtract(1001,1000),$multiply(100,100))"  
6  }
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the binary multiplication function. Parameter A in the function is the

local parameter **test**, and parameter B is **1**. For details about how to set local parameters, see [Local Parameters](#).

- **if condition**

As shown in the following figure, the target value of the **if** condition is the binary multiplication function. Parameter A in the function is **100**, and parameter B is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

- **for loop interrupt condition**

As shown in the following figure, the target value of the **for** loop interrupt condition is the binary multiplication function. Parameters A and B in the function are **100**.

6.4.11.4 Binary Division Operation

Function Name

`$divide(intA, intB, intC)`

Function Description

Implements the division operation between parameter A and parameter B. C is the precision value. Parameters A, B, and C support the following types:

- Numbers
- Local parameters
- Binary operations
 - Division operation without precision: For exact division, the value is the number of reserved digits. For inexact division, the value is rounded off with six decimal places by default.
 - Division operation with precision: The precision value is an integer ranging from 1 to 6. For exact division, the reserved decimal places must be in the precision range. For inexact division, the value is rounded off with the specified number of decimal places.

Application Scenarios

The binary division function can be used in the following scenarios for API automation:

- Request URL

- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the binary division function. Parameter A in the function is **1000** and parameter B is **100**.

Name	Value
test	<code>\$divide(1000,100)</code>

- As shown in the following figure, the value of **test** in the request URL is the binary division function with a precision value. Parameter A in the function is **1**, parameter B is **3**, and the precision value is **5**.

Name	Value
test	<code>\$divide(1,3,5)</code>

- Request header

As shown in the following figure, the value of **divide** in the request header is the binary division function. Parameter A in the function is the binary addition operation **\$add(1000,1000)**, and parameter B is **-1**.

Name	Value
Content-Type	application/json
divide	<code>\$divide(\$add(1000,1000),-1)</code>

As shown in the following figure, the value of **divide** in the request header is the binary division function with a precision value. Parameter A in the function is **1**, parameter B is **-3**, and parameter C is the global environment parameter **\$\$scale**.

Name	Value
divide	<code>\$divide(1,-3,\$\$scale)</code>

- Request body

As shown in the following figure, the binary division function is used in the request body. Parameter A in the function is the binary subtraction operation **\$subtract(1001,1000)**, and parameter B is the binary multiplication operation **\$multiply(100,100)**.

```
1 {  
2   "add": "$add($multiply(100,100),$divide(1000,100))",  
3   "subtract": "$subtract($divide(1000,100),$add(1000,1000))",  
4   "multiply": "$multiply($add(1000,1000),$subtract(1001,1000))",  
5   "divide": "$divide($subtract(1001,1000),$multiply(100,100))"  
6 }
```

As shown in the following figure, the binary division function with a precision value is used in the request body. Parameter A in the function is **1**, parameter B is **3**, and parameter C is the global environment parameter **`\${scale}`**.

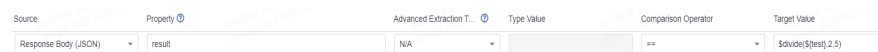
```
1 {  
2   "divide": "$divide(1,3,`${scale}`)"  
3 }  
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the binary division function. Parameter A in the function is the local parameter **test**, and parameter B is **1**. For details about how to set local parameters, see [Local Parameters](#).

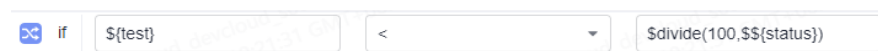


As shown in the following figure, the target value of the checkpoint property **result** is the binary division function with a precision value. Parameter A in the function is the local parameter **test**, parameter B is **2**, and parameter C is **5**. For details about how to set local parameters, see [Local Parameters](#).



- if condition

As shown in the following figure, the target value of the **if** condition is the binary division function. Parameter A in the function is **1**, and parameter B is the environment variable **status**. Parameter C is the local parameter **localScale**. For details about how to set local parameters, see [Local Parameters](#).



As shown in the following figure, the target value of the **if** condition is the binary division function with a precision value. Parameter A in the function is **1** and parameter B is **3**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

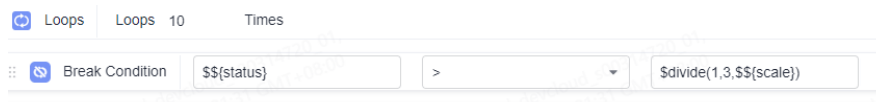


- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the binary division function. Parameter A in the function is **1000**, and parameter B is **100**.



As shown in the following figure, the target value of the **for** loop interrupt condition is the binary division function with a precision value. Parameter A in the function is **1**, parameter B is **3**, and parameter C is the global environment parameter **\$\$scale**.



6.4.11.5 Obtaining the Current Timestamp

Function Name

\$timestamp()

Function Description

Obtains the total number of milliseconds from 1970-01-01 00:00:00 to the current time.

Application Scenarios

The function for obtaining the current timestamp can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the function for generating the current timestamp.

Name	Value
test	\$timestamp()

- Request header

As shown in the following figure, the value of **time** in the request header is the function for generating the current timestamp.

Name	Value
Content-Type	application/json
time	\$timestamp()

- Request body

As shown in the following figure, the request body uses the function for generating the current timestamp.

```
1 {  
2   "time": "$timestamp()  
3 }  
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the function for generating the current timestamp.

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		==	\$timestamp()

- if condition

As shown in the following figure, the target value of the **if** condition is the function for generating the current timestamp.

if	\$(test)	<	\$timestamp()
----	----------	---	---------------

- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the function for generating the current timestamp.

Loops	Loops	Times
Break Condition	\$(status)	> \$timestamp()

6.4.11.6 Obtaining a Specified Timestamp

Function Name

\$getTimeBeforeHour(*doubleA*)

Function Description

Obtains the timestamp of the time that is *A* hours ago. The timestamp is the total number of milliseconds from 1970-01-01 00:00:00 to the specified time.

Parameter *A* in the function supports the following types:

- Numbers
- Local parameters

- Other built-in functions

Application Scenarios

The function for obtaining the specified timestamp can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the function for obtaining the specified timestamp. Parameter A in the function is **3**.

Name	Value
test	\$getTimeBeforeHour(3)

- Request header

As shown in the following figure, the value of **time** in the request header is the function for obtaining the specified timestamp. Parameter A in the function is **3**.

Name	Value
Content-Type	application/json
time	\$getTimeBeforeHour(3)

- Request body

As shown in the following figure, the request body uses the function for obtaining the specified timestamp. Parameter A in the function is the binary addition operation **\$add(2,2)**.

```
1  {  
2    "time": "$getTimeBeforeHour($add(2,2))"  
3  }  
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the function for obtaining the specified timestamp. Parameter A in

the function is the local parameter **test**. For details about how to set local parameters, see [Local Parameters](#).



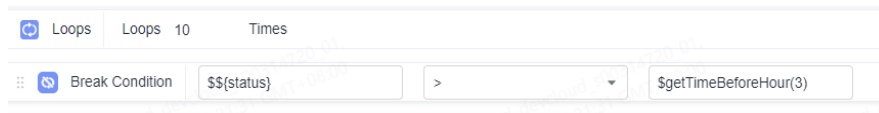
- **if** condition

As shown in the following figure, the target value of the **if** condition is the function for obtaining the specified timestamp. Parameter A in the function is the environment variable **time**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).



- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the function for obtaining the specified timestamp. Parameter A in the function is **3**.



6.4.11.7 Converting a Date into a Timestamp

Function Name

`$dateFormat(String A)`

Parameter Description

String A: date and time value. The following formats are supported:

- yyyy-MM-dd HH:mm:ss or MM-dd-yyyy HH:mm:ss
- yyyy MM dd HH:mm:ss or MM dd yyyy HH:mm:ss
- yyyy.MM.dd HH:mm:ss or MM.dd.yyyy HH:mm:ss
- yyyy/MM/dd HH:mm:ss or MM/dd/yyyy HH:mm:ss

Function Description

Converts a string into a timestamp. The timestamp is the total number of milliseconds from 1970-01-01 00:00:00 to the specified time.

Parameter A in the function supports the following types:

- Date and time in the format listed in [Parameter Description](#)
- Local parameters
- Other built-in functions

Application Scenarios

The date-to-timestamp function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the date-to-timestamp conversion function. Parameter A in the function is the environment parameter **date**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

Name	Value
test	\$dateFormat(\$\$(date))

- Request header

As shown in the following figure, the value of **date** in the request header is the date-to-timestamp conversion function. Parameter A in the function is the environment parameter **date**.

Name	Value
Content-Type	application/json
date	\$dateFormat(\$\$(date))

- Request body

As shown in the following figure, the request body uses the date-to-timestamp conversion function. Parameter A in the function is **2020.09.11 11:00:00**.

```
1  {  
2    "date": "$dateFormat(2020.09.11 11:00:00)"  
3  }  
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the date-to-timestamp function. Parameter A in the function is the environment parameter **test**. For details about how to set local parameters, see [Local Parameters](#).

Source	Property	Advanced Extraction T.	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		==	\$dateFormat(\$\$(date))

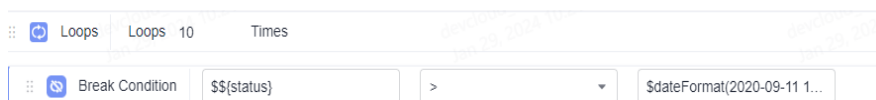
- **if** condition

As shown in the following figure, the target value of the **if** condition is the date-to-timestamp function. Parameter A in the function is **2020-09-11 11:00:00**.



- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interruption condition is the date-to-timestamp function. Parameter A in the function is **2020-09-11 11:00:00**.



6.4.11.8 Converting a Timestamp into a Date

Function Name

`$timestamp_format(String A, String B)`

Parameter Description

- *String A*: timestamp to be converted. The value is a numeric string containing a maximum of 20 digits. You can also use the built-in function **\$timestamp()** to obtain the current timestamp.
- *String B*: date and time value, consisting of the year, month, day, hour, minute, second, and millisecond. Where,
 - Year: represented by letter "y" and consists of 4 characters.
 - Month: represented by letter "M" and consists of 1 to 2 characters.
 - Day: represented by letter "d" and consists of 1 to 2 characters.
 - Hour: represented by letter "H" and consists of 0 to 2 characters.
 - Minute: represented by letter "m" and consists of 0 to 2 characters.
 - Second: represented by letter "s" and consists of 0 to 2 characters.
 - Millisecond: represented by letter "S" and consists of 3 characters.
- **Table 6-5** Letters and corresponding parameter types

Letter	Date/Time Element	Type	Example
G	Era identifier	Text	AD
y	Year	Year	1996, 96
M	Month in a year	Month	July, Jul, 07
w	Week number in a year	Number	27
W	Week number in a month	Number	2

Letter	Date/Time Element	Type	Example
D	Day number in a year	Number	189
d	Day number in a month	Number	10
F	Week number in a month	Number	2
E	Day in a week	Text	Tuesday, Tue
a	am/pm flag	Text	PM
H	Hour number in a day (0-23)	Number	0
k	Hour number in a day (1-24)	Number	24
K	Hour number in am/pm (0-11)	Number	0
h	Hour number in am/pm (1-12)	Number	12
m	Minute number in an hour	Number	30
s	Second number in a minute	Number	55
S	Millisecond	Number	978
z	Time zone abbreviation	Text	PST, EST
Z	Time zone offset	Text	+800, -0530

 NOTE

1. Each letter has its own meaning and is case sensitive.
2. If one of "H", "m", "s" is **0**, the other two must also be **0**.

A date and time value can contain spaces, hyphens (-), slashes (/), and colons (:), but not escape characters, such as \n. Common formats include but are not limited to the following:

- yyyy-MM-dd HH:mm:ss
- yyyyMMddHHmmss
- yyyyMMddHHmmssSSS
- yyyy-M-d H:m:s
- MM-dd-yyyy HH:mm:ss

- MM/dd/yyyy HH/mm/ss
- MM/d/yyyy H/mm/ss
- MM/d/yyyy H/mm/ss SSS
- yyyyMMdd SSS
- yyyy-MM-dd HH:mm:ss SSS
- yyyy-MM-dd HH:mm:ss
- yyyy-MM-dd HH:mm
- yyyy-MM-dd HH
- yyyy-MM-dd
- yyyy-MM
- yyyy
- yy
- MM-dd HH
- MM-dd
- MM
- dd
- HH:mm:ss SSS
- HH:mm:ss
- HH:mm
- HH
- mm
- mm:ss
- ss
- SSS

Function Description

Converts a timestamp into a date in the corresponding format. The timestamp is the total number of milliseconds from 1970-01-01 00:00:00 to the specified time.

Parameter A in the function supports the following types:

- Date and time in the format listed in [Parameter Description](#)
- Environment parameters
- Local parameters
- Other built-in functions

Parameter B in the function supports the following types:

- Date and time in the format listed in [Parameter Description](#)
- Environment parameters
- Local parameters
- Other built-in functions

Application Scenarios

The timestamp-to-date function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **date** in the request URL is the timestamp-to-date conversion function. Parameter A in the function is the built-in function **\$timestamp()** for obtaining the current timestamp. For details, see [Obtaining the Current Timestamp](#). Parameter B is **yyyy-MM-dd HH:mm:ss**.

Name	Value
date	\$timestamp_format(\$timestamp(), yyyy-MM-dd HH:mm:ss)

- Request header

As shown in the following figure, the value of **date** in the request header is the timestamp-to-date conversion function. Parameter A in the function is the environment parameter **date**, and parameter B is **yyyyMMddHHmmss**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

Name	Value
test	\$timestamp_format(\$date, yyyyMMddHHmmss)

- Request body

As shown in the following figure, the request body uses the timestamp-to-date conversion function. Parameter A in the function is **123456789**, and parameter B is **yyyyMMddHHmmssSSS**.

```
1 {  
2   "date": "$timestamp_format(123456789, yyyyMMddHHmmssSSS)"  
3 }  
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the timestamp-to-date function. Parameter A in the function is the built-in function **\$timestamp()** for obtaining the current timestamp, and parameter B is **MM/dd/yyyy HH/mm/ss**.

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		Equals	\$timestamp_format(\$timestamp(), MM/dd/yyyy HH/mm/ss)

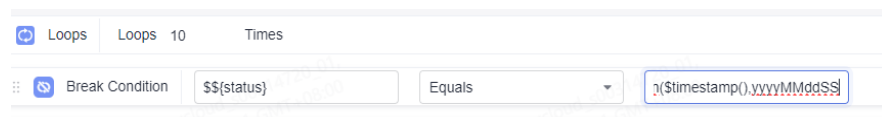
- **if** condition

As shown in the following figure, the target value of the **if** condition is the timestamp-to-date function. Parameter A in the function is the built-in function **\$timestamp()** for obtaining the current timestamp, and parameter B is **MM/d/yyyy H/mm/ss SSS**.



- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the timestamp-to-date function. Parameter A in the function is the built-in function **\$timestamp()** for obtaining the current timestamp, and parameter B is **yyyyMMdd SSS**.



6.4.11.9 Timestamp Addition and Subtraction Operations

Function Name

`$timeStampCalculation(longA, StringB)`

Parameter Description

- *longA*: timestamp in milliseconds.
- *StringB*: time difference. The value is an integer plus a letter ("d": day; "h": hour; "s": second). For example, **1d** indicates that one day is added to the specified timestamp, and **-1d** indicates that one day is subtracted.

Function Description

Implements the addition and subtraction operations between parameter A of the long type and parameter B of the string type. Parameters A and B support the following types:

- Value in the format listed in [Parameter Description](#)
- Local parameters
- Other built-in functions

Application Scenarios

Timestamp addition and subtraction operations can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property

- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the timestamp addition and subtraction function. Parameter A in the function is **1607939485441**, and parameter B is **1d**.

Name	Value
test	\$timeStampCalculation(1607939485441,1d)

- Request header

As shown in the following figure, the value of **time** in the request header is the timestamp addition and subtraction function. Parameter A in the function is the date-to-timestamp conversion function **\$dateFormat(2020.09.11 11:00:00)**, and parameter B is **-86400s**.

Name	Value
Content-Type	application/json
TIME	\$timeStampCalculation(\$dateFormat(2020.09.11 11:00:00),-86400s)

- Request body

As shown in the following figure, the request body uses the timestamp addition and subtraction function. Parameter A in the function is **\$dateFormat(2020.09.11 11:00:00)**, and parameter B is **1d**.

```
1 {  
2   "time": "$timeStampCalculation($dateFormat(2020.09.11 11:00:00),1d)"  
3 }  
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the timestamp addition and subtraction function. Parameter A in the function is the environment parameter **time**, and parameter B is **-24h**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		==	StampCalculation(\$time,-24h)

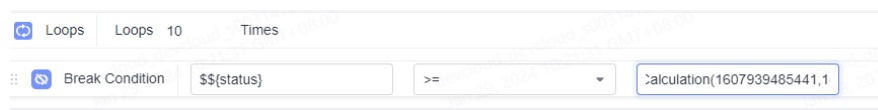
- **if** condition

As shown in the following figure, the target value of the **if** condition is the timestamp addition and subtraction function. Parameter A in the function is the environment variable **status**, and parameter B is **1d**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

if	\$(test)	<=	StampCalculation(\$\$(date),1d)
----	----------	----	---------------------------------

- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the timestamp addition and subtraction function. Parameter A in the function is **1607939485441**, and parameter B is **1d**.



6.4.11.10 Generating Base64 Encoding

Function Name

`$encode_base64(StringA)`

Parameter Description

StringA: string to be encoded. The value contains a maximum of 256 bytes and supports the following special characters: !*() ;@&=+\$/?#[]-.~%<>_{}^`.

Function Description

Performs Base64 encoding on strings. Parameter A supports the following types:

- Value in the format listed in [Parameter Description](#)
- Environment parameters
- Local parameters
- Other built-in functions

Application Scenarios

The Base64 encoding generation function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the Base64 encoding generation function. Parameter A in the function is the string **abc123**.



- Request header

As shown in the following figure, the value of **Accept-Encoding** in the request header is the Base64 encoding generation function. Parameter A in the function is the string **abc123**.

Name	Value
Content-Type	application/json
Accept-Encoding	\$encode_base64(abc123)

- Request body

As shown in the following figure, the request body uses the Base64 encoding generation function. Parameter A in the function is **\$uuid()**.

```
1 {  
2   "encoding": "$encode_base64($uuid())"  
3 }  
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the Base64 encoding generation function. Parameter A in the function is the local parameter **test**. For details about how to set local parameters, see [Local Parameters](#).

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		Contains	\$encode_base64(\${test})

- if condition

As shown in the following figure, the target value of the **if** condition is the Base64 encoding generation function. Parameter A in the function is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

if	\${test}	Contains	\$encode_base64(\${status})
----	----------	----------	-----------------------------

- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the Base64 encoding generation function. Parameter A in the function is the string **abc123**.

Loops	Loops	10	Times
Break Condition	`\${status}	Does not contain	\$encode_base64(abc123)

6.4.11.11 Generating SHA512 Encoding

Function Name

\$sha512(*StringA*)

Parameter Description

StringA: string to be encoded. The value contains a maximum of 256 bytes and supports the following special characters: !*() ;@&=+\$,/?#[]~.%<>_{}`^.

Function Description

Performs SHA512 encoding on strings. Parameter A supports the following types:

- Value in the format listed in [Parameter Description](#)
- Environment parameters
- Local parameters
- Other built-in functions

Application Scenarios

The SHA512 encoding generation function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the SHA512 encoding generation function. Parameter A in the function is the string **abc123**.

Name	Value
test	\$sha512(abc123)

- Request header

As shown in the following figure, the value of **Accept-Encoding** in the request header is the SHA512 encoding generation function. Parameter A in the function is the string **abc123**.

Name	Value
Content-Type	application/json
Accept-Encoding	\$sha512(abc123)

- Request body

As shown in the following figure, the request body uses the SHA512 encoding generation function. Parameter A in the function is **\$uuid()**.

```
1  {  
2    "encoding": "$sha512($uuid())"  
3  }  
4  
5
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the SHA512 encoding generation function. Parameter A in the

function is the local parameter **test**. For details about how to set local parameters, see [Local Parameters](#).

- **if** condition

As shown in the following figure, the target value of the **if** condition is the SHA512 encoding generation function. Parameter A in the function is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the SHA512 encoding generation function. Parameter A in the function is the string **abc123**.

6.4.11.12 Generating SHA256 Encoding

Function Name

`$sha256(StringA)`

Parameter Description

StringA: string to be encoded. The value contains a maximum of 256 bytes and supports the following special characters: !*() ; : @ & = + \$, / ? # [] - . ~ % < > _ { } ` ^ .

Function Description

Performs SHA256 encoding on strings. Parameter A supports the following types:

- Value in the format listed in [Parameter Description](#)
- Environment parameters
- Local parameters
- Other built-in functions

Application Scenarios

The SHA256 encoding generation function can be used in the following scenarios for API automation cases:

- Request URL
- Request header
- Request body

- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the SHA256 encoding generation function. Parameter A in the function is the string **abc123**.

Name	Value
test	\$sha256(abc123)

- Request header

As shown in the following figure, the value of **Accept-Encoding** in the request header is the SHA256 encoding generation function. Parameter A in the function is the string **abc123**.

Name	Value
Content-Type	application/json
Accept-Encoding	\$sha256(abc123)

- Request body

As shown in the following figure, the request body uses the SHA256 encoding generation function. Parameter A in the function is **\$uuid()**.

```
1  {  
2    "encoding": "$sha256($uuid())"  
3  }  
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the SHA256 encoding generation function. Parameter A in the function is the local parameter **test**. For details about how to set local parameters, see [Local Parameters](#).

Source	Property	Advanced E...	Type Value	Comparison Op...	Target Value
Response Bo...	result	N/A		Contains	\$sha256(\${test})

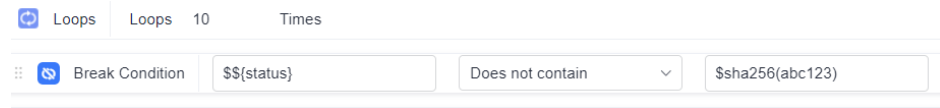
- **if** condition

As shown in the following figure, the target value of the **if** condition is the SHA256 encoding generation function. Parameter A in the function is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

if	\${test}	Contains	\$sha256(status)
----	----------	----------	------------------

- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the SHA256 encoding generation function. Parameter A in the function is the string **abc123**.



6.4.11.13 Generating an MD5 Hash Value

Function Name

`$md5(StringA)`

Parameter Description

StringA: string to be encoded. The value contains a maximum of 256 bytes and supports the following special characters: !*() ;@&=+\$,/?#[]-.%<>_{}^.

Function Description

Converts a string into an MD5 hash value. Parameter A supports the following types:

- Value in the format listed in [Parameter Description](#)
- Environment parameters
- Local parameters
- Other built-in functions

Application Scenarios

The MD5 hash function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the MD5 hash value generation function. Parameter A in the function is the string **abc123**.



- Request header

As shown in the following figure, the value of **Accept-Encoding** in the request header is the MD5 hash value generation function. Parameter A in the function is the string **abc123**.

Name	Value
Content-Type	application/json
Accept-Encoding	\$md5(abc123)

- Request body

As shown in the following figure, the request body uses the MD5 hash value generation function. Parameter A in the function is **\$uuid()**.

```
1 {  
2   "encoding": "$md5($uuid())"  
3 }  
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the MD5 hash value generation function. Parameter A in the function is the local parameter **test**. For details about how to set local parameters, see [Local Parameters](#).

Source	Property ID	Advanced Extraction T.	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		Contains	\$md5(\$test)

- if condition

As shown in the following figure, the target value of the **if** condition is the MD5 hash value generation function. Parameter A in the function is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

if	if	\$test	Contains	\$md5(\$status)
----	----	--------	----------	-----------------

- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the MD5 hash value generation function. Parameter A in the function is the string **abc123**.

Loops	Loops	10	Times
Break Condition	Break Condition	Does not contain	\$md5(abc123)

6.4.11.14 Generating a Random Number in a Specified Range

Function Name

\$random_int(*intA*, *intB*)

Function Description

Generates a random number within the range between parameter A and parameter B. CodeArts TestPlan can generate random numbers containing a maximum of 10 digits, that is, the value range is [-9999999999, +9999999999].

Parameters A and B support the following types:

- Numbers
- Environment parameters
- Local parameters
- Other built-in functions

Application Scenarios

The function for generating random numbers within a specified range can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the function for generating random numbers within a specified range. Parameter A in the function is **1**, and parameter B is **100**.

Name	Value
test	\$random_int(1,100)

- Request header

As shown in the following figure, the value of **number** in the request header is the function for generating random numbers within a specified range. Parameter A in the function is **1**, and parameter B is **100**.

Name	Value
Content-Type	application/json
number	\$random_int(1,100)

- Request body

As shown in the following figure, the request body uses the function for generating random numbers within a specified range. Parameter A in the function is the binary addition operation **\$add(5,5)**. Parameter B is the binary multiplication operation **\$multiply(5,5)**.

```
1  {
2    "number": "$random_int($add(5,5), $multiply(5,5))"
3  }
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the function for generating random numbers within a specified range. Parameter A in the function is **1**, and parameter B is the local parameter **test**. For details about how to set local parameters, see [Local Parameters](#).



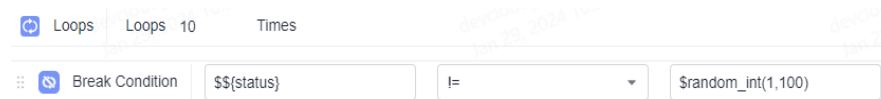
- if condition

As shown in the following figure, the target value of the **if** condition is the function for generating random numbers within a specified range. Parameter A in the function is **1**, and parameter B is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).



- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the function for generating random numbers within a specified range. Parameter A in the function is **1**, and parameter B is **100**.



6.4.11.15 Generating a Number with Specified Decimal Places

Function Name

\$numberFormat(double number,int A)

Function Description

Generates a number with a specified number of decimal places.

Parameter *number* supports the following types:

- Numbers
- Environment parameters
- Local parameters
- Other built-in functions

Parameter *A* is the number of digits you want to reserve, which ranges from 0 to 20. For example, if the function **\$numberFormat(3.14,0)** is used, the decimal places will not be retained and the number 3 will be returned.

Application Scenarios

This function can be used in the following parts of an API automation test case. (Do not use this function for environment parameters. Otherwise, what is input will be output.)

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of the **test** parameter in the request URL is the function for generating a number with a specified number of decimal places. Parameters *number* and *A* in the function are **3.14** and **0**, respectively.

Request Parameters:

Name	Value
test	\$numberFormat(3.14,0)

- Request header

As shown in the following figure, the value of the **number** parameter in the request header is the function for generating a number with a specified number of decimal places. Parameter *number* in the function is a function for random generation of decimals within a specified range, and parameter *A* is **0**.

Header:

Name	Value
Content-Type	application/json
number	\$numberFormat(\$randomDecimal(1, 100, 2),0)

6.4.11.16 Generating a Random String of a Specified Length

Function Name

\$random_string(*intA*)

Function Description

Generates a random string of a specified length. Parameter *A* supports the following types:

- Numbers
- Environment parameters

- Local parameters
- Other built-in functions

Application Scenarios

The function for generating random strings of a specified length can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the function for generating random strings of a specified length. Parameter A in the function is **3**.

Name	Value
test	\$random_string(3)

- Request header

As shown in the following figure, the value of the **string** parameter in the request header is the function for generating random strings of a specified length. Parameter A in the function is **3**.

Name	Value
Content-Type	application/json
string	\$random_string(3)

- Request body

As shown in the following figure, the request body uses the function for generating random strings of a specified length. Parameter A in the function is the binary addition operation **\$add(2,2)**.

```
1 {  
2   "string": "$random_string($add(2,2))"  
3 }  
4  
5
```

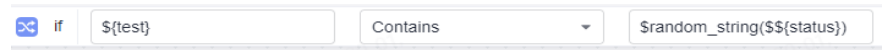
- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the function for generating random strings of a specified length. Parameter A in the function is the local parameter **test**. For details about how to set local parameters, see [Local Parameters](#).

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		Contains	\$random_string(test)

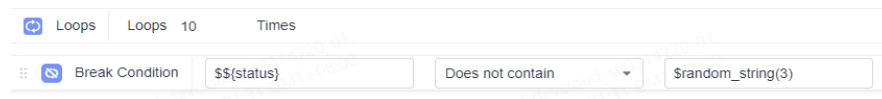
- **if** condition

As shown in the following figure, the target value of the **if** condition is the function for generating random strings of a specified length. Parameter A in the function is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).



- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interruption condition is the function for generating random strings of a specified length. Parameter A in the function is **3**.



6.4.11.17 Generating a Random Decimal in a Specified Range

Function Name

`$randomDecimal(double A, double B, int C)`

Function Description

Generates a random decimal with *C* decimal places within the range between parameter A and parameter B. CodeArts TestPlan can generate random decimals with up to 9 decimal places, within the range of `[-9999999999.999999999, +9999999999.999999999]`.

Parameters A and B support the following types:

- Decimals or integers
- Environment parameters
- Local parameters
- Other built-in functions

Parameter C supports the following types:

- Positive integers
- Environment parameters
- Local parameters
- Other built-in functions

If *C* is **0**, the function generates a random integer.

Application Scenarios

The function for generating random decimals within a specified range can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the function for generating random decimals within a specified range. Parameter A in the function is **1**, parameter B is **100**, and parameter C is **2**.

Request Parameters:

Name	Value
test	\$randomDecimal(1, 100, 2)

- Request header

As shown in the following figure, the value of **number** in the request header is the function for generating random decimals within a specified range. Parameter A in the function is **1**, parameter B is **100**, and parameter C is **2**.

Header:

Name	Value
Content-Type	application/json
number	\$randomDecimal(1, 100, 2)

- Request body

As shown in the following figure, the request body uses the function for generating random decimals within a specified range. Parameter A in the function is the binary addition operation **\$add(1,1)**. Parameter B is the binary multiplication operation **\$multiply(10,10)**.

Parameter C is **2**.

```
1 {  
2   "number": "$randomDecimal($add(1,0), $multiply(10,10), int 2)"  
3 }  
4
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the function for generating random decimals within a specified range. Parameter A in the function is **1**, parameter B is the local parameter **test**, and parameter C is **2**. For details about how to set local parameters, see [Local Parameters](#).

Source	Property	Advanced Extraction Type	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		==	\$randomDecimal(1, 5, test), 2)

- **if** condition

As shown in the following figure, the target value of the **if** condition is the function for generating random decimals within a specified range. Parameter A in the function is **1**, parameter B is the environment variable **status**, and parameter C is **2**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).

if	`\${test}`	==	domDecimal(1, `\${status}`, 2)
----	------------	----	--------------------------------

- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the function for generating random decimals within a specified range. Parameter A in the function is **1**, parameter B is **100**, and parameter C is **2**.

Loops	Loops 10	Times
Break Condition	`\${status}`	!= \$randomDecimal(1, 100, 2)

6.4.11.18 Generating a UUID

Function Name

\$uuid()

Function Description

Generates a random string.

Application Scenarios

The UUID generation function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the UUID generation function.

Name	Value
test	\$uuid()

- Request header

As shown in the following figure, the value of **time** in the request header is the UUID generation function.

Name	Value
Content-Type	application/json
string	\$uuid()

- Request body

As shown in the following figure, the request body uses the UUID generation function.

```
1  {  
2    "string": "$uuid()  
3  }  
4  
5
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the UUID generation function.

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		Contains	\$uuid()

- if condition

As shown in the following figure, the target value of the **if** condition is the UUID generation function.

if	`\${test}`	Contains	\$uuid()
----	------------	----------	----------

- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the UUID generation function.

Loops	Loops	Times
Break Condition	`\${status}`	Does not contain \$uuid()

6.4.11.19 Custom URL Encoding

Function Name

\$urlEncoder(*String A*)

Parameter Description

String A: parameter to be encoded. It can contain up to 2,000 characters. UTF-8 is used for encoding.

Function Description

This function is used for custom URL encoding of API automation test case parameters.

Application Scenarios

This function can be used in the following parts of an API automation test case. (Do not use the function for environment parameters. Otherwise, input will be output.)

- Request URL
- Request header
- Request Body
- Checkpoint property
- Response extraction
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the custom URL encoding function. Parameter A in the function is the string **abc123Test**.

Request Parameters:

Name	Value
test	\$urlEncoder(abc123Test)

- Request header

As shown in the following figure, the value of **Accept-Encoding** in the request URL is the custom URL encoding function. Parameter A in the function is the string **abc123Test**.

Header:

Name	Value
Content-Type	application/json
Accept-Encoding	\$urlEncoder(abc123Test)

- Request body

As shown in the following figure, the request body uses the custom URL encoding function. Parameter A in the function is the function for generating random strings of a specified length **\$random_string(10)**.

```
1  {  
2    "test": "$urlEncoder($random_string(10))"  
3  }  
4
```

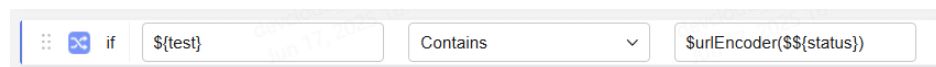
- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the custom URL encoding function. Parameter A in the function is the local parameter **test**. For details about how to set local parameters, see [Local Parameters](#).



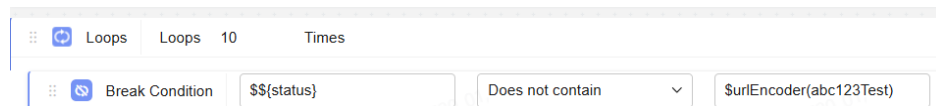
- if condition

As shown in the following figure, the target value of the **if** condition is the custom URL encoding function. Parameter A in the function is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).



- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the custom URL encoding function. Parameter A in the function is the string **abc123Test**.



6.4.11.20 URL Decoding

Function Name

`$urlDecoder(String A)`

Parameter Description

String A: parameter to be decoded. It can contain up to 2,000 characters. UTF-8 is used for decoding.

Function Description

After a custom URL encoding is performed on parameters in a request, the parameters need to be transcoded again so that they can be received and parsed by the requested service. URL decoding can be performed on parameters in API automation test cases.

Application Scenarios

This function can be used in the following parts of an API automation test case. (Do not use the function for environment parameters. Otherwise, input will be output.)

- Request URL

- Request header
- Request body
- Checkpoint property
- Response extraction
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the URL decoding function. Parameter A in the function is the encoded string **abc123%E6%B5%8B%E8%AF%95**.

Request Parameters:

Name	Value
test	\$urlDecoder(abc123%E6%B5%8B%E8%AF%95)

- Request header

As shown in the following figure, the value of **Accept-Encoding** in the request header is the custom URL decoding function. Parameter A in the function is the encoded string **abc123%E6%B5%8B%E8%AF%95**.

Header:

Name	Value
Content-Type	application/json
Accept-Encoding	\$urlDecoder(abc123%E6%B5%8B%E8%AF%95)

- Request body

As shown in the following figure, the request body uses the URL decoding function. Parameter A in the function is the custom URL encoding function **\$urlEncoder(abc123Test)**.

```
1  {
2    "test": "$urlDecoder($urlEncoder(abc123Test))"
3  }
4
```

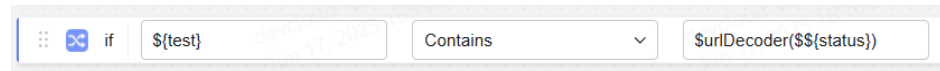
- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the custom URL decoding function. Parameter A in the function is the local parameter **test**. For details about how to set local parameters, see [Local Parameters](#).

Check whether the response is the expected result. ⓘ					
Source	Property ⓘ	Advanced Extraction ⓘ	Type Value	Comparison Operator	Target Value
⋮ Response Body (JSON) ▾	result	N/A ▾		Equals ▾	\$urlDecoder(\${test})

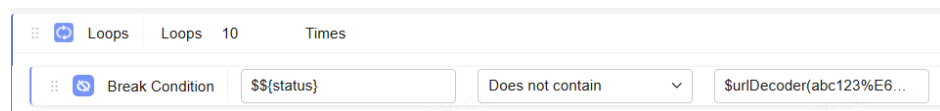
- **if** condition

As shown in the following figure, the target value of the **if** condition is the custom URL decoding function. Parameter A in the function is the environment variable **status**. For details about how to set environment parameters, see [Setting Environment Parameters of an API Script](#).



- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the custom URL decoding function. Parameter A in the function is the string **abc123%E6%B5%8B%E8%AF%95**.



6.4.11.21 Obtaining an Array via Reverse Index

Function Name

`$getReverseltem(StringA, intB)`

Parameter Description

- **StringA**: array or list element path of the response body or response header.
- **intB**: subscript of the array in reverse order. **0** indicates the last array, and **2** indicates the last but one array.

Function Description

Obtains an array from a response body or header based on the reverse index.

Application Scenarios

The built-in function for obtaining arrays during response extraction based on the reverse index can be used for API automation.

Example

As shown in the following figure, the property value of the response body parameter **item** is the function for obtaining arrays via reverse Index. Parameter A in the function is the response body property **result**, and parameter B is **0**.



6.4.11.22 Obtaining the Element Values of an Array via Reverse Index

Function Name

`$getReverseltem(StringA, StringB, intC)`

Parameter Description

- *StringA*: array or list element path of the response body or response header.
- *StringB*: property name of the array object.
- *intC*: subscript of the array in reverse order. **0** indicates the last array, and **2** indicates the last but one array.

Function Description

Obtains a specified element value of an array from a response body or header based on the reverse index.

Application Scenarios

The built-in function for obtaining the element values of an array during response extraction based on the reverse index can be used for API automation.

Example

As shown in the following figure, the property value of the response parameter **name** is the element value of the array obtained via reverse index. Parameter A in the function is the response body property **result**, parameter B is the parameter name **name** in the last N+1 array in the result, and parameter C is **0**.

Output Parameter	Source	Property
name	Response Body (JSON)	\$getReverseItem(result,name,0)

6.4.11.23 Converting Uppercase Letters into Lowercase Letters

Function Name

\$toLower(*String A*)

Parameter Description

- *String A*: string to be converted into lowercase letters.

Parameter A supports the following types:

- Strings
- Local parameters

Function Description

Converts all characters in the specified input string into lowercase characters.

Application Scenarios

The function for converting uppercase letters into lowercase letters can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the function for converting uppercase letters into lowercase letters. Parameter A in the function is **TEST**.

Name	Value
test	\$toLower(TEST)

- Request header

As shown in the following figure, the value of **lower** in the request header is the function for converting uppercase letters into lowercase letters. Parameter A in the function is **Test**.

Name	Value
Content-Type	application/json
lower	\$toLower(Test)

- Request body

As shown in the following figure, the function for converting uppercase letters into lowercase letters is used. Parameter A in the function is **Test**.

```
1 {  
2   "id":1,  
3   "lower":"$toLower(TEST)"  
4 }  
5
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the function for converting uppercase letters into lowercase letters. Parameter A in the function is **Test**.

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		==	\$toLower(Test)

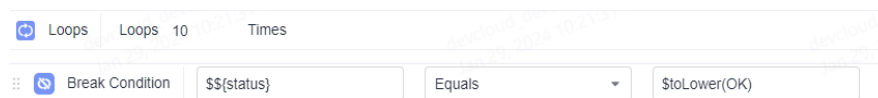
- **if** condition

As shown in the following figure, the target value of the **if** condition is the function for converting uppercase letters into lowercase letters. Parameter A in the function is **AAAAA**.

 if	\$toLower(AAAAA)	Equals	aaaaa
--	------------------	--------	-------

- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the function for converting uppercase letters into lowercase letters. Parameter A in the function is **OK**.



6.4.11.24 Converting Lowercase Letters into Uppercase Letters

Function Name

\$toUpper(*String A*)

Parameter Description

- *String A*: string to be converted into uppercase letters.

Parameter A supports the following types:

- Strings
- Local parameters

Function Description

Converts all characters in the specified input string into uppercase characters.

Application Scenarios

The function for converting lowercase letters into uppercase letters can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the function for converting lowercase letters into uppercase letters. Parameter A in the function is **test**.



- Request header

As shown in the following figure, the value of **upper** in the request header is the function for converting lowercase letters into uppercase letters. Parameter A in the function is **Test**.

Name	Value
Content-Type	application/json
upper	\$toUpper(Test)

- Request body

As shown in the following figure, the function for converting lowercase letters into uppercase letters is used. Parameter A in the function is **Test**.

```
Request Body:
1 {
2   "ip": 1,
3   "upper": "$toUpper(Test)"
4 }
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the function for converting lowercase letters into uppercase letters. Parameter A in the function is **Test**.

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		Contains	\$toUpper(Test)

- if condition

As shown in the following figure, the target value of the **if** condition is the function for converting lowercase letters into uppercase letters. Parameter A in the function is **aaaaa**.

if	\$toUpper(aaaaa)	Equals	AAAAA
----	------------------	--------	-------

- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the function for converting lowercase letters into uppercase letters. Parameter A in the function is **ok**.

Loops	Loops 10	Times	
Break Condition	\$\$ {status}	Equals	\$toUpper(ok)

6.4.11.25 Concatenating Strings

Function Name

`$strConCat(String A, String B)`

Parameter Description

- String A*: string 1.
- String B*: string 2.

Parameters A and B support the following types:

- Strings
- Local parameters

Function Description

Concatenates strings 1 and 2 into a new string.

Application Scenarios

The string concatenation function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of the **test** parameter in the request URL is the string concatenation function. Parameters A and B in the function are **abc** and **123** respectively.

Name	Value
test	<code>\$strConcat(abc, 123)</code>

- Request header

As shown in the following figure, the value of **number** in the request header is the string concatenation function. Parameter A in the function is **00000**, and parameter B is the environment parameter **`\${number}`**.

Name	Value
Content-Type	application/json
number	<code>\$strConcat(00000, `\${number}`)</code>

- Request body

As shown in the following figure, the string concatenation function is used in the request body. Parameter A in the function is the environment parameter **`\${info}`**, and parameter B is the environment parameter **`\${number}`**.

```
1 {  
2   "id":1,  
3   "address":"$strConcat(`${info}`, `${number}`)"  
4 }  
5
```

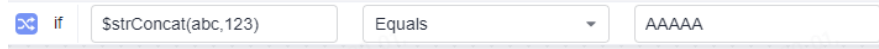
- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the string concatenation function. Parameter A in the function is the local parameter **`\${str1}`**, and parameter B is the local parameter **`\${str2}`**.

Source	Property	Advanced Extraction T.	Type	Value	Comparison Operator	Target Value
Response Body (JSON)	result		N/A		Equals	<code>\$strConcat(`\${str1}`, `\${str2}`)</code>

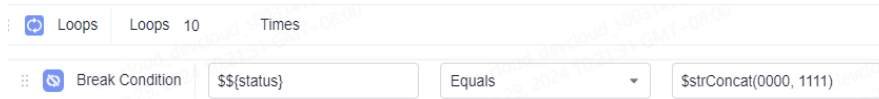
- **if** condition

As shown in the following figure, the target value of the **if** condition is the string concatenation function. Parameter A in the function is **abc**, and parameter B is **123**.



- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the string concatenation function. Parameter A in the function is **0000**, and parameter B is **1111**.



6.4.11.26 Concatenating Strings of Multiple Parameters

Function Name

\$strJoin(StringA,StringB,StringC.....)

Parameter Description

Adds multiple parameters to a function. The parameters should be of the following types:

- Strings
- Local parameters
- Environment parameters
- Other built-in functions

Function Description

Concatenates multiple strings into a new string.

Application Scenarios

This function can be used in the following parts of an API automation test case. (Do not use the function for environment parameters. Otherwise, what is input will be output.)

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of the **test** parameter in the request URL is the function for concatenating strings of multiple parameters. Parameter A in the function is **abc**, parameter B is **123**, and parameter C is the environment parameter **\$\${number}**.

Request Parameters:

Name	Value
test	\$strJoin(abc, 123, \$\$ {number})

- Request header

As shown in the following figure, the value of the **number** parameter in the request header is the function for concatenating strings of multiple parameters. Parameters A, B, and C in the function are the local parameters **\${str1}**, **\${str2}**, and **\${str3}**, respectively.

Header:

Name	Value
Content-Type	application/json
number	\$strJoin(\${str1}, \${str2}, \${str3})

- Request body

As shown in the following figure, the function for concatenating strings of multiple parameters is used in the request body. Parameters A and B in the function are the environment parameters **\$\${info}** and **\$\${number}**, respectively.

```
1 {  
2   "id": 1,  
3   "address": "$strJoin($${info}, $${number})"  
4 }  
5
```

- Checkpoint property

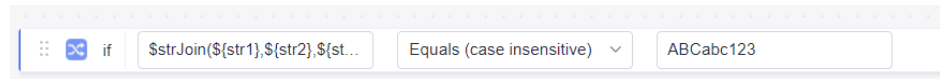
As shown in the following figure, the target value of the checkpoint property **result** is the function for concatenating strings of multiple parameters. Parameters A, B, and C in the function are the local parameters **\${str1}**, **\${str2}**, and **\${str3}**, respectively.

Check whether the response is the expected result. ⓘ					
Source	Property ⓘ	Advanced Extractio... ⓘ	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		Equals (case insensitive)	\$strJoin(\${str1}, \${str2}, \${str3})

- if condition

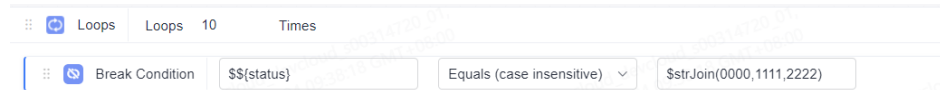
As shown in the following figure, the target value of the **if** condition is the function for concatenating strings of multiple parameters. Parameters A, B,

and C in the function are the local parameters **`${str1}`**, **`${str2}`**, and **`${str3}`**, respectively.



- **for** loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the function for concatenating strings of multiple parameters. Parameters A, B, and C in the function are **0000**, **1111**, and **2222**, respectively.



6.4.11.27 Cutting Strings

Function Name

`$strCut(String A, int B, int C)`

Parameter Description

- *String A*: original string to be cut.
- *int B*: start subscript of the string to be cut, starting from **0**.
- *int C*: end subscript of the string to be cut.

Parameters A, B, and C support the following types:

- Strings
- Local parameters

Function Description

Obtains the value string of a specified element and cuts it to a new string.

Application Scenarios

The string cutting function can be used in the following scenarios for API automation:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the string cutting function. Parameter A in the function is the environment parameter **\$\$user**, parameter B is **2**, and parameter C is **4**.

Name	Value
name	\$strCut(\$\${user}, 2, 4)

- Request header

As shown in the following figure, the value of **name** in the request header is the string cutting function. Parameter A in the function is the environment parameter **\$\$user**, parameter B is **2**, and parameter C is **4**.

Name	Value
Content-Type	application/json
name	\$strCut(\$\${user}, 2, 4)

- Request body

As shown in the following figure, the request body uses the string cutting function. Parameter A in the function is the environment parameter **\$\$user**, parameter B is **2**, and parameter C is **4**.

```
1  {
2      "id": "$toUpper($${a})",
3      "name": "$strCut($${user}, 2, 4)"
4  }
5
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the string cutting function. Parameter A in the function is the environment parameter **\$\$info**, parameter B is **2**, and parameter C is **5**.

Source	Property	Advanced Extraction T.	Type Value	Comparison Operator	Target Value
Response Body (JSON)	result	N/A		==	\$strCut(\$\${info}, 2, 5)

- if condition

As shown in the following figure, the target value of the **if** condition is the string cutting function. Parameter A in the function is **abcdef**, parameter B is **2**, and parameter C is **4**.

if	\$strCut(abcdef, 2, 4)	Equals	cd
----	------------------------	--------	----

- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interruption condition is the string cutting function. Parameter A in the function is the environment parameter **\$\$test**, parameter B is **2**, and parameter C is **4**.

Loops	Loops	Times
Break Condition	\$\${info}	Equals \$strCut(\$\${test}, 2, 4)

6.4.11.28 Obtaining String Length

Function Name

`$strlen(String A)`

Parameter Description

- *String A*: original string whose length is requested.

Parameter A supports the following types:

- Strings
- Local parameters

Function Description

Obtains the length of a specified string.

Application Scenarios

The string length function can be used in the following scenarios for API automation cases:

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the following figure, the value of **test** in the request URL is the string length function that uses the local parameter **`${test}`**.

Name	Value
test	<code>\$strlen(\${test})</code>

- Request header

As shown in the following figure, the value of **test** in the request header is the string length function whose parameter A is the local parameter **`${test}`**.

Name	Value
Content-Type	application/json
test	<code>\$strlen(\${test})</code>

- Request body

As shown in the following figure, the request body uses the string length function whose parameter A is the local parameter **`${test}`**.

```
1  {
2    |    "String": "${strLen(${test})}"
3  }
```

- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the string length function whose parameter A is the environment parameter **`$${info}`**.

Source	Property	Advanced E...	Type Value	Comparison Op...	Target Value
Response Bo...	result	N/A		==	\$strLen(\$\${info})

- if condition

As shown in the following figure, the target value of the **if** condition is the string length function uses the local parameter **`${test}`**.

if	\$strLen(\${test})	==	10
----	--------------------	----	----

- for loop interrupt condition

As shown in the following figure, the target value of the **for** loop interrupt condition is the string length function uses the local parameter **`${test}`**.

Loops	Loops	10	Times
Break Condition	\$\${info}	==	\$strLen(\${test})

6.4.11.29 Extracting a JSON Object

Function Name

`$extractingData(String json, String jsonPath, String mode)`

Parameter Description

- json*: Accepts a local parameter, environment parameter, or built-in function. Must be in JSON format.
- jsonPath*: Accepts a string, local parameter, environment parameter, or built-in function.
- mode*: Accepts **jayway** or **fastJson**.

Function Description

Extracts a JSON object from the API response for subsequent API test steps, and assigns the object value to other parameters.

Application Scenarios

This function can be used in the following parts of an API automation test case. (Do not use the function for environment parameters. Otherwise, what is input will be output.)

- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the URL request in [Figure 2](#), the function for extracting a JSON object is used to extract the value of **id** returned by the body parameter (as shown in [Figure 1](#)) and assign the value to **group_id**.

Figure 6-1 Response

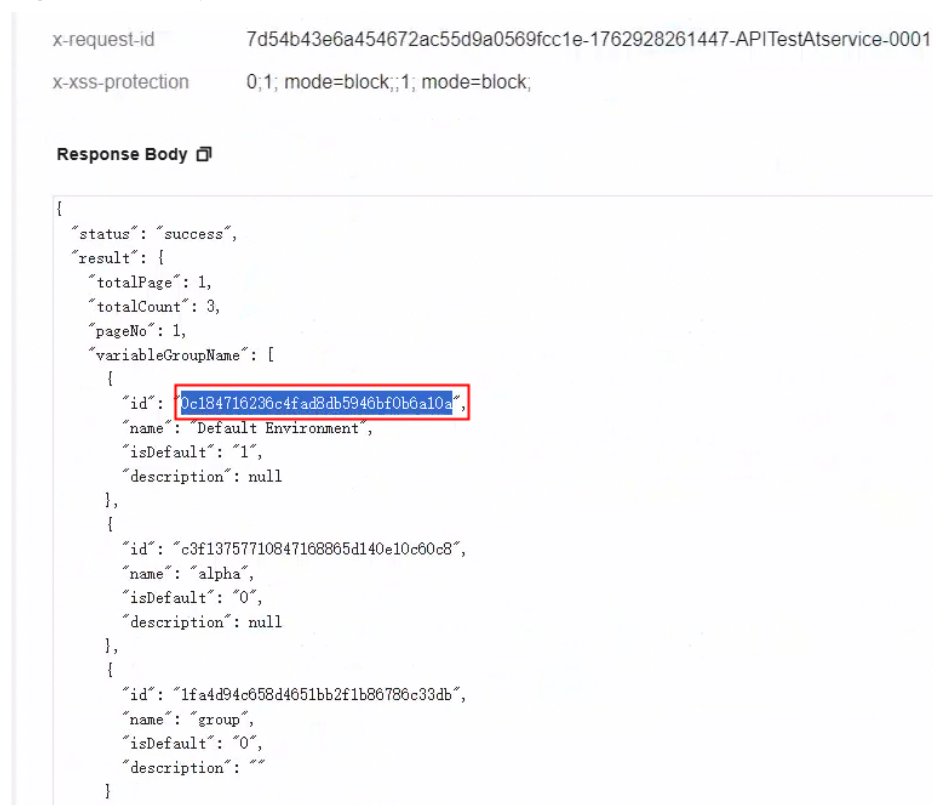


Figure 6-2 Extracting a JSON object

Request Uri:

GET http://

Request Parameters:

Name	Value	Description
page_no	1	
page_size	0	
group_id	\$extractingData(\$body).\$id.jayway)	

After saving and running the script, you can view the details in the **execution history**.

Details

Step Name	Json (451 ms)
URL	https://dables?page_no=1&page_size=0&group_id=0c184716236c4fad8db5946bf0b6a10a
Request Type	GET
HTTP Version	HTTP 1.1
Started	2025/11/12 14:17:41 724 GMT+08:00
Ended	2025/11/12 14:17:42 175 GMT+08:00
Status	Successful
Response Code	200

6.4.11.30 Replacing a String

Function Name

\$strReplace(String str, String old, String replace, boolean isRegular)

Parameter Description

- str*: Accepts a string, local parameter, environment parameter, or built-in function.
- old*: Accepts a string, local parameter, environment parameter, or built-in function.
- replace*: Accepts a string, local parameter, environment parameter, or built-in function.
- replace*: Accepts **true** or **false** to indicate whether to use a regular expression for replacement. If it is set to **true**, the *old* parameter accepts regex.

Function Description

Replaces a URL request parameter value or part of the value and assigns the value to the *str* parameter.

Application Scenarios

This function can be used in the following parts of an API automation test case. (Do not use the function for environment parameters. Otherwise, what is input will be output.)

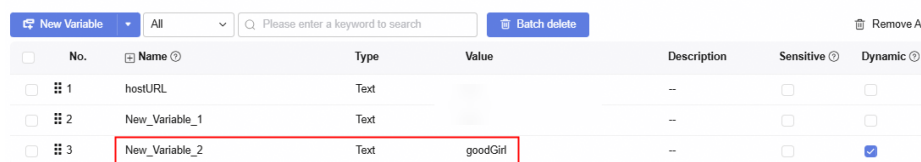
- Request URL
- Request header
- Request body
- Checkpoint property
- **if** condition
- **for** loop interrupt condition

Example

- Request URL

As shown in the URL request in [Figure 2](#), the string replacement function is used in the request body to replace the value of the environment parameter **New_Variable_2** and assign the new value to **keyword**.

Figure 6-3 Old environment parameter value



No.	Name	Type	Value	Description	Sensitive	Dynamic
1	hostURL	Text		--	☐	☐
2	New_Variable_1	Text		--	☐	☐
3	New_Variable_2	Text	goodGirl	--	☐	☒

Figure 6-4 Replacing the environment parameter value

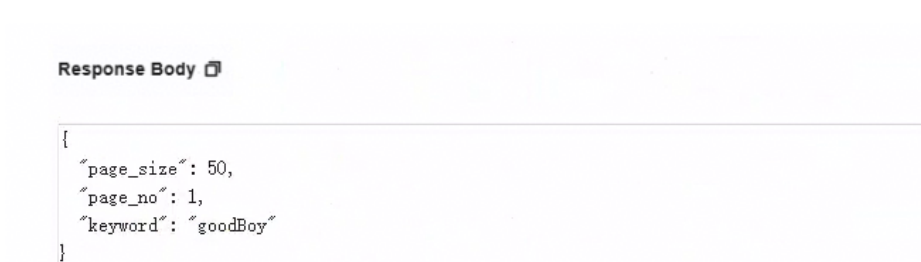


Request Body:

☐ none ☐ form-data ☐ x-www-form-urlencoded ☐ binary ☒ raw

```
{\"page_size\":50,\"page_no\":1,\"keyword\":\"${Replace:${New_Variable_2},Girl,Boy,true)}
```

After saving and running the script, you can view the details in the **execution history**.



Response Body

```
{  \"page_size\": 50,  \"page_no\": 1,  \"keyword\": \"goodBoy\"}
```

6.4.12 System Keywords

6.4.12.1 Overview

Frequent operations of automated API tests are encapsulated into keywords for more efficient API test case writing. For details about the keywords, see [Table 6-6](#).

Table 6-6 System keywords

Category	Keyword Set
Authentication	• GetIAMToken
Database operation	• MySQLQuery • MySQLUpdate • MySQLInsert • MySQLDelete • OpenGaussQuery • OpenGaussUpdate • OpenGaussInsert • OpenGaussDelete • PostgreSQLQuery • PostgreSQLUpdate • PostgreSQLInsert • PostgreSQLDelete • MongoDBQuery • MongoDBInsert • MongoDBUpdate • MongoDBDelete
Middleware	• RedisGet • RedisSet • OBSWrite • OBSDelete • OBSQuery • KafkaProducer • KafkaConsumer
Protocol	• TCP • UDP • WSConnect • WSRequest • WSWriteOnly • WSReadOnly • WSDisconnect • DubboClient

6.4.12.2 GetIAMToken

Overview

This system keyword is used for obtaining an IAM user token through username or password authentication. A token is an access credential issued to an IAM user. It carries their identity and permissions. When calling APIs of IAM and other cloud services, you can use this system keyword to obtain an IAM user token.

Parameter Description

Parameter	Mandatory	Type	Default Value	Description
IAM Token URL	Yes	String	https://iam.myhuaweicloud.com/v3/auth/tokens	IAM endpoint. This API can be called using both global and regional endpoints.
Domain Name	Yes	String	<Empty>	IAM account or tenant name. When a Huawei account is used for login, the tenant (account) name is the same as the username. For details about how to obtain an account name, see Obtaining Account, IAM User, Group, Project, Region, and Agency Information .
User Name	Yes	String	<Empty>	IAM username. For details about how to obtain a username, see Obtaining Account, IAM User, Group, Project, Region, and Agency Information .
Password	Yes	String	<Empty>	IAM user login password. (The login password is personal information and must be defined as sensitive in the environment parameters.)
Region ID	No	String	<Empty>	Region ID, for example, cn-north-1 . For details about how to obtain a region ID, see Obtaining Account, IAM User, Group, Project, Region, and Agency Information .

Default Checkpoint

Name	Expected Value
Result	Success

Default Response Extraction

Name	Extraction Variable	Description
IAM_TOKEN	X-Subject-Token	User token character

Response

- **Status: success**

Parameter	Type	Description
Body	Response body of the IAM API	Response body of the IAM API

- **Example response**

```
{
  "token": {
    "catalog": [],
    "expires_at": "2020-01-04T09:05:22.701000Z",
    "issued_at": "2020-01-03T09:05:22.701000Z",
    "methods": [
      "password"
    ],
    "project": {
      "domain": {
        "id": "d78cbac186b744899480f25bd022f...",
        "name": "IAMDomain"
      },
      "id": "aa2d97d7e62c4b7da3ffdfc11551f...",
      "name": "cn-north-1"
    },
    "roles": [
      {
        "id": "0",
        "name": "te_admin"
      },
      {
        "id": "0",
        "name": "op_gated_OBS_file_protocol"
      },
      {
        "id": "0",
        "name": "op_gated_Video_Campus"
      }
    ],
    "user": {
      "domain": {
        "id": "d78cbac186b744899480f25bd022f...",
```

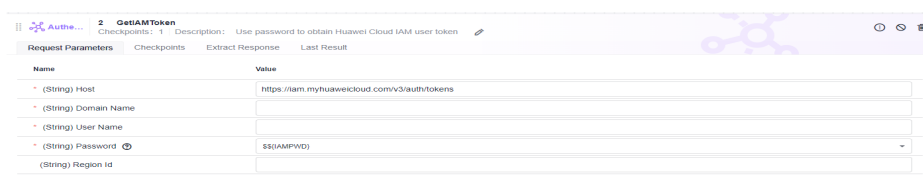
```
    "name": "IAMDomain"
  },
  "id": "7116d09f88fa41908676fdd4b039e...",
  "name": "IAMUser",
  "password_expires_at": ""
}
},
"X-Subject-Token": "MIlatAYJKoZIhvcNAQcCollapTCCGqECAQExDTALB..."
}
```

Example

- Obtaining an IAM user token

NOTE

The **Password** field needs to be configured as a sensitive parameter in the variable. Select a value from the drop-down list box.

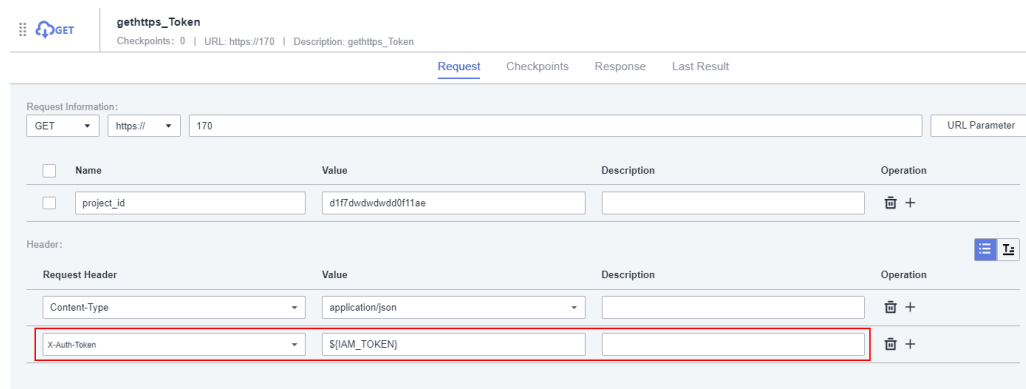


Name	Value
(String) Host	https://iam.myhuaweicloud.com/v3/auth/tokens
(String) Domain Name	
(String) User Name	
(String) Password	\$(PASSWORD)
(String) Region Id	

- Referencing a token in an API

NOTE

IAM_TOKEN is the reserved default response extraction and can be directly referenced in the request.



Name	Value	Description	Operation
project_id	d1f7dwdwdwd0f11ae		+ -

Request Header	Value	Description	Operation
Content-Type	application/json		+ -
X-Auth-Token	\$(IAM_TOKEN)		+ -

6.4.12.3 MySQLQuery

This system keyword is used for **select** operations on MySQL database. A maximum of 100 result records can be queried in the system.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a Huawei Cloud RDS instance, bind an EIP to the instance and configure its security group rules to allow incoming access. For details, see Using MySQL CLI to Connect to an Instance Through a Public Network .
Port	Yes	Integer	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL query statement

MySQLQuery Response

Status: success

Parameter	Type	Description
[<i>Array elements</i>]	Array of row objects	Structure of the SQL query result list

Parameter Description

Parameter	Type	Description
field1	String	Database field 1
field2	Integer	Database field 2
...	String	Database field <i>n</i>

MySQLQuery Response Example

```
[  
  {  
    "name": "Sam",  
    "id": "efdb403066474ab08836b9eeaaa23bca",  
    "age": 18  
  },  
  {
```

```
"name": "Sue",  
"id": "g582b0d966611486f918bedb9c711b14",  
"age": 20  
}  
]
```

MySQLQuery Usage Example

- Query the database table. A user data list is returned based on the system keyword. You can determine the return based on the service.

The screenshot shows the 'Request Parameters' tab of the 'MySQLQuery' configuration. It lists several parameters with their values:

Name	Value
(String) Ip	\$(db_ip)
(Integer) Port	\$(db_port)
(String) DB Name	\$(db_name)
(String) User Name	\$(db_username)
(String) Password	\$(db_password)
(String) Sql	select * from serviceinfo limit 3

Determine the result in the returned list.

The screenshot shows the 'Checkpoints' tab of the 'MySQLQuery' configuration. It displays a table for defining checkpoints:

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value	operation
Result					Successful	+ ↓
Response value (JSON)	[0] name	N/A		Equals	cjd	+ ↑

- Query and collect user data. A user data list is returned.

The screenshot shows the 'Request Parameters' tab of the 'MySQLQuery' configuration. It lists several parameters with their values:

Name	Value
(String) Ip	\$(db_ip)
(Integer) Port	\$(db_port)
(String) DB Name	\$(db_name)
(String) User Name	\$(db_username)
(String) Password	\$(db_password)
(String) Sql	select count(*) as count from serviceinfo;

Determine the result in the response extraction.

The screenshot shows the 'Checkpoints' tab of the 'MySQLQuery' configuration. It displays a table for defining checkpoints:

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value	operation
Result					Successful	+ ↓
Response value (JSON)	[0] count	N/A		==	1180	+ ↑

6.4.12.4 MySQLUpdate

This system keyword is used for modification operations on MySQL database. It is applicable to statements, such as **INSERT**, **UPDATE**, and **DELETE**, which have no returns.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a Huawei Cloud RDS instance, bind an EIP to the instance and configure its security group rules to allow incoming access. For details, see Using MySQL CLI to Connect to an Instance Through a Public Network .
Port	Yes	Integer	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL statement, such as UPDATE , which has no returns

MySQLUpdate Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by the SQL statement

MySQLUpdate Usage Example

The screenshot shows the 'Request Parameters' tab of the 'MySQLUpdate' test case. It lists six parameters with their types and values:

Name	Value
(String) Ip	\$\$db_ip
(Integer) Port	\$\$db_port
(String) DB Name	\$\$db_name
(String) User Name	\$\$db_username
(String) Password	\$\$db_password
(String) Sql	UPDATE 'atmock','serviceinfo' SET 'name'='cjd2','createTime'='2020-05-04 21:00:27','baseId'='1.0','tag'='1.0','port'=NULL,{'

Determine the number of affected rows.

The screenshot shows the 'Checkpoints' tab of the 'MySQLUpdate' test case. It displays a table for defining checkpoints:

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value	operation
Result					Successful	+ ↓ □
Response value (JSON)	affectedRowCount		N/A	==	1	+ ↑ □

MySQLUpdate Response Example

```
{  
  "affected_row_count": 3  
}
```

6.4.12.5 MySQLInsert

This system keyword is used for data insertion on MySQL database.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a Huawei Cloud RDS instance, bind an EIP to the instance and configure its security group rules to allow incoming access. For details, see Using MySQL CLI to Connect to an Instance Through a Public Network .
Port	Yes	Integer	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL statement, such as INSERT , which has no returns

MySQLInsert Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by the SQL statement

MySQLInsert Response Example

```
{
  "affected_row_count": 1
}
```

MySQLInsert Usage Example

The screenshot shows the 'Request Parameters' tab of the MySQLInsert configuration. It lists several parameters with their types and values:

Name	Value
(String) Ip	\$(mysql_ip)
(Integer) Port	\$(mysql_port)
(String) DB Name	\$(mysql_db)
(String) User Name	\$(mysql_user)
(String) Password	\$(mysql_password)
(String) Sql	INSERT INTO company(id,name,age,address,salary,join_date)VALUES(0,'apitest',32,'China',20000,'2001-07-13 00:00:00');

Determine the number of affected rows.

The screenshot shows the 'Checkpoints' tab of the MySQLInsert configuration. It displays a table for defining checkpoints:

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value	operation
Result					Successful	+ ↓ ⌵
Response value (JSON)	affected_row_count	N/A		==	1	+ ↑ ⌴

6.4.12.6 MySQLDelete

This system keyword is used for deletion operations on MySQL database.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a Huawei Cloud RDS instance, bind an EIP to the instance and configure its security group rules to allow incoming access. For details, see Using MySQL CLI to Connect to an Instance Through a Public Network.

Parameter	Mandatory	Type	Description
Port	Yes	Integer	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL statement, such as DELETE , which has no returns

MySQLDelete Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by the SQL statement

MySQLDelete Response Example

```
{
  "affected_row_count": 1
}
```

MySQLDelete Usage Example

1 MySQLDelete	
Checkpoints: 1 Description: Delete operation on MySQL database	
Request Parameters Checkpoints Extract Response Last Result	
Name	Value
* (String) Ip	\$(mysql_ip)
* (Integer) Port	\$(mysql_port)
* (String) DB Name	\$(mysql_db)
* (String) User Name	\$(mysql_user)
* (String) Password	\$(mysql_password)
* (String) Sql	delete from company where name='apitest'

Determine the number of affected rows.

2 MySQLDelete	
Checkpoints: 2 Description: Delete operation on MySQL database	
Request Parameters Checkpoints Extract Response Last Result	
Retries: Retry Interval(ms)	
Check whether the response is the expected result.	
Source	Property
Result	affected_row_count
Response value (JSON)	affected_row_count
Advanced Extraction T.	N/A
Type Value	
Comparison Operator	==
Target Value	1
operation	

6.4.12.7 OpenGaussQuery

This system keyword is used for **select** operations on openGauss database. A maximum of 100 result records can be queried in the system.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address For a cloud-based GaussDB instance, bind an EIP to the instance and ensure that the security group policy of the instance port is enabled to allow access. For details, see Using gsql to Connect to an Instance .
Port	Yes	Integer	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL query statement

OpenGaussQuery Response

Status: success

Parameter	Type	Description
[Array elements]	Array of row objects	Structure of the SQL query result list

Parameter Description

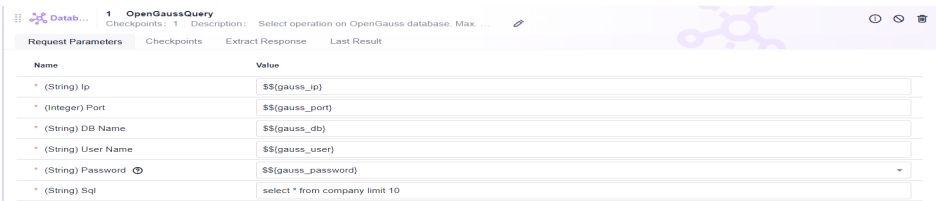
Parameter	Type	Description
field1	String	Database field 1
field2	Integer	Database field 2
...	String	Database field <i>n</i>

OpenGaussQuery Response Example

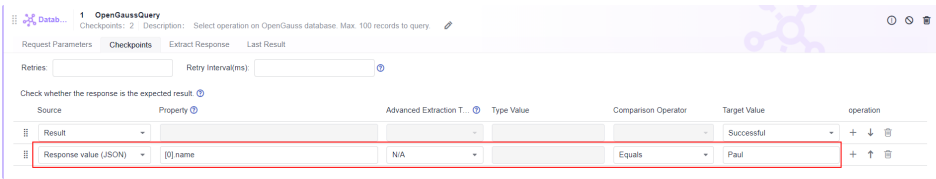
```
[
  {
    "name": "Sam",
    "id": "efdb403066474ab08836b9eeaaa23bca",
    "age": 18
  },
  {
    "name": "Sue",
    "id": "g582b0d966611486f918bedb9c711b14",
    "age": 20
  }
]
```

OpenGaussQuery Usage Example

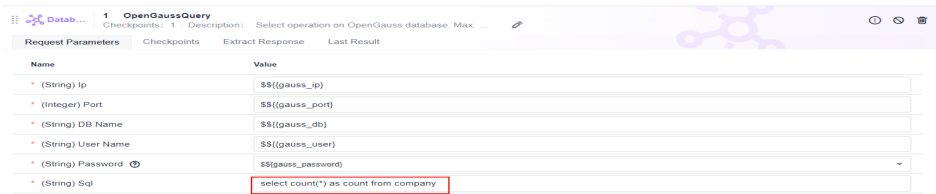
Query the database table. A user data list is returned based on the system keyword. You can determine the return based on the service.



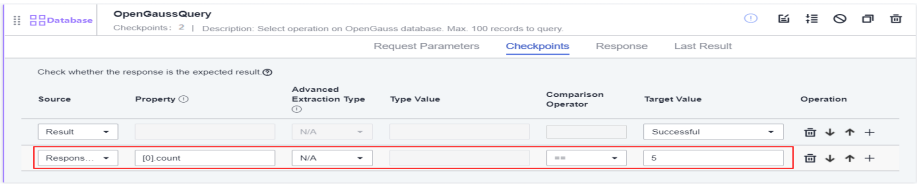
Determine the result in the returned list.



Query and collect user data. A user data list is returned.



Determine the result in the response extraction.



6.4.12.8 OpenGaussUpdate

This system keyword is used for modification operations on openGauss database.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a cloud-based GaussDB instance, bind an EIP to the instance and ensure that the security group policy of the instance port is enabled to allow access. For details, see Using gsql to Connect to an Instance .
Port	Yes	Integer	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL statement, such as UPDATE , which has no returns

OpenGaussUpdate Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by the SQL statement

OpenGaussUpdate Response Example

```
{
  "affected_row_count": 1
}
```

OpenGaussUpdate Usage Example

The screenshot shows the 'OpenGaussUpdate' configuration window. The 'Request Parameters' tab is active, displaying a table with the following parameters:

Name	Value
(String) Ip	\$\$gauss_ip
(Integer) Port	\$\$gauss_port
(String) DB Name	\$\$gauss_db
(String) User Name	\$\$gauss_user
(String) Password	\$\$gauss_password
(String) Sql	update company set salary=20000 where name='apitest'

Determine the number of affected rows.

The screenshot shows the 'OpenGaussUpdate' configuration window with the 'Checkpoints' tab active. It displays a table for defining checkpoints:

Source	Property	Advanced Extraction T.	Type Value	Comparison Operator	Target Value	operation
Result					Successful	
Response value (JSON)	affected_row_count	N/A		==	1	

6.4.12.9 OpenGaussInsert

This system keyword is used for data insertion on openGauss database.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a cloud-based GaussDB instance, bind an EIP to the instance and ensure that the security group policy of the instance port is enabled to allow access. For details, see Using gsqsl to Connect to an Instance .
Port	Yes	Integer	Database port

Parameter	Mandatory	Type	Description
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL statement, such as INSERT , which has no returns

OpenGaussInsert Response

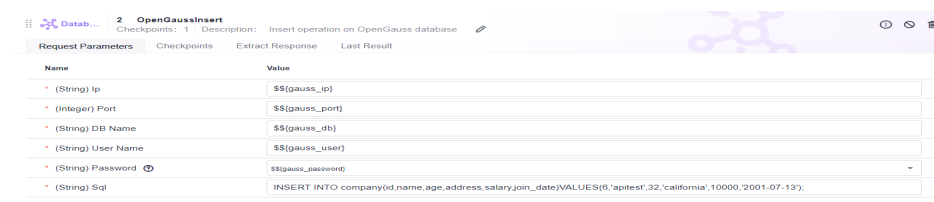
Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by the SQL statement

OpenGaussInsert Response Example

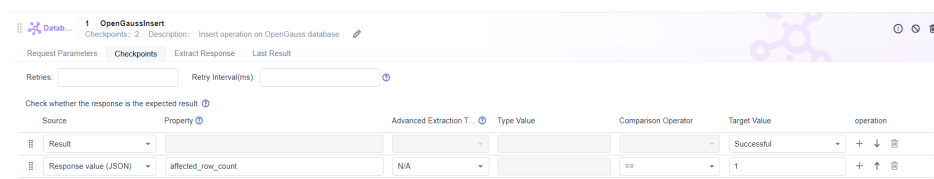
```
{
  "affected_row_count": 1
}
```

OpenGaussInsert Usage Example



Name	Value
ip (String)	\$(gauss_ip)
Port (Integer)	\$(gauss_port)
DB Name (String)	\$(gauss_db)
User Name (String)	\$(gauss_user)
Password (String)	\$(gauss_password)
Sql (String)	INSERT INTO company(id,name,age,address,salary,join_date)VALUES(6,'apitest',32,'california',10000,'2001-07-13');

Determine the number of affected rows.



Source	Property	Advanced Extraction T.	Type Value	Comparison Operator	Target Value	operation
Result	affected_row_count	N/A		Successful	1	

6.4.12.10 OpenGaussDelete

This system keyword is used for deletion operations on openGauss database.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a cloud-based GaussDB instance, bind an EIP to the instance and ensure that the security group policy of the instance port is enabled to allow access. For details, see Using gsql to Connect to an Instance .
Port	Yes	Integer	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL statement, such as DELETE , which has no returns

OpenGaussDelete Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by the SQL statement

OpenGaussDelete Response Example

```
{
  "affected_row_count": 1
}
```

OpenGaussDelete Usage Example

Name	Value
(String) Ip	\$\$gauss_ip
(Integer) Port	\$\$gauss_port
(String) DB Name	\$\$gauss_db
(String) User Name	\$\$gauss_user
(String) Password	\$\$gauss_password
(String) Sql	delete from company where name='apitest'

Determine the number of affected rows.

Source	Property	Advanced Extraction T.	Type Value	Comparison Operator	Target Value	operation
Result	affected_row_count	N/A		==	1	

6.4.12.11 PostgreSQLQuery

This system keyword is used for **select** operations on PostgreSQL database. A maximum of 100 result records can be queried in the system.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a Huawei Cloud RDS instance, bind an EIP to the instance and ensure that the security group policy of the instance port is enabled to allow access. For details, see the connection methods .
Port	Yes	String	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username

Parameter	Mandatory	Type	Description
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL query statement

PostgreSQLQuery Response

Status: success

Parameter	Type	Description
[<i>Array elements</i>]	Array of row objects	Structure of the SQL query result list

Parameter Description

Parameter	Type	Description
field1	String	Database field 1
field2	Integer	Database field 2
...	String	Database field <i>n</i>

PostgreSQLQuery Response Example

```
[
  {
    "name": "Sam",
    "id": "efdb403066474ab08836b9eeaaa23bca",
    "age": 18
  },
  {
    "name": "Sue",
    "id": "g582b0d966611486f918bedb9c711b14",
    "age": 20
  }
]
```

PostgreSQLQuery Usage Example

Query the database table. A user data list is returned based on the system keyword. You can determine the return based on the service.

Name	Value
(String) Ip	\$(postgresql_ip)
(Integer) Port	\$(postgresql_port)
(String) DB Name	\$(postgresql_db)
(String) User Name	\$(postgresql_username)
(String) Password	\$(postgresql_password)
(String) Sql	select * from company

Determine the result in the returned list.

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value	operation
Result	[0] name	N/A		Equals	Paul	

Query and collect user data. A user data list is returned.

Name	Value
(String) Ip	\$(postgresql_ip)
(Integer) Port	\$(postgresql_port)
(String) DB Name	\$(postgresql_db)
(String) User Name	\$(postgresql_username)
(String) Password	\$(postgresql_password)
(String) Sql	select * from company

Determine the result in the response extraction.

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value	operation
Result	[0] count	N/A		==	5	

6.4.12.12 PostgreSQLUpdate

This system keyword is used for modification operations on PostgreSQL database.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a Huawei Cloud RDS instance, bind an EIP to the instance and ensure that the security group policy of the instance port is enabled to allow access. For details, see the connection methods .

Parameter	Mandatory	Type	Description
Port	Yes	Integer	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL statement, such as UPDATE , which has no returns

PostgreSQLUpdate Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by the SQL statement

PostgreSQLUpdate Response Example

```
{
  "affected_row_count": 1
}
```

PostgreSQLUpdate Usage Example

Name	Value
* (String) ip	\$(postgresql_ip)
* (Integer) Port	\$(postgresql_port)
* (String) DB Name	\$(postgresql_db)
* (String) User Name	\$(postgresql_user)
* (String) Password	\$(postgresql_password)
* (String) Sql	update company set salary=20000 where name="apitest"

Determine the number of affected rows.

Source	Property	Advanced Extraction T.	Type Value	Comparison Operator	Target Value	operation
Result				Successful		
Response value (JSON)	affected_row_count	N/A		==	1	

6.4.12.13 PostgreSQLInsert

This system keyword is used for data insertion on PostgreSQL database.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a Huawei Cloud RDS instance, bind an EIP to the instance and ensure that the security group policy of the instance port is enabled to allow access. For details, see the connection methods .
Port	Yes	Integer	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL statement, such as INSERT , which has no returns

PostgreSQLInsert Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by the SQL statement

PostgreSQLInsert Response Example

```
{
  "affected_row_count": 1
}
```

PostgreSQLInsert Usage Example

The screenshot shows the 'Request Parameters' tab of the 'PostgreSQLInsert' test case. It lists several parameters with their values:

Name	Value
*(String) Ip	\$(postgresql_ip)
*(Integer) Port	\$(postgresql_port)
*(String) DB Name	\$(postgresql_db)
*(String) User Name	\$(postgresql_user)
*(String) Password	\$(postgresql_password)
*(String) Sql	INSERT INTO company(id,name,age,address,salary,join_date)VALUES(6,'apitest',32,'California')

Determine the number of affected rows.

The screenshot shows the 'Checkpoints' tab of the 'PostgreSQLInsert' test case. It displays a table for defining checkpoints:

Source	Property	Advanced Extraction T.	Type Value	Comparison Operator	Target Value	operation
Result					Successful	+ - & ☒
Response value (JSON)	affected_row_count	N/A		=	1	+ - & ☒

6.4.12.14 PostgreSQLDelete

This system keyword is used for deletion operations on PostgreSQL database.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address. For a Huawei Cloud RDS instance, bind an EIP to the instance and ensure that the security group policy of the instance port is enabled to allow access. For details, see the connection methods .
Port	Yes	Integer	Database port
DB Name	Yes	String	Database instance name
User Name	Yes	String	Username

Parameter	Mandatory	Type	Description
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Sql	Yes	String	SQL statement, such as DELETE , which has no returns

PostgreSQLDelete Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by the SQL statement

PostgreSQLDelete Response Example

```
{
  "affected_row_count": 1
}
```

PostgreSQLDelete Usage Example

Name	Value
(String) Ip	\$\$postgresql_ip
(Integer) Port	\$\$postgresql_port
(String) DB Name	\$\$postgresql_db
(String) User Name	\$\$postgresql_user
(String) Password	\$\$postgresql_password
(String) Sql	delete from company where name='apitest'

Determine the number of affected rows.

Source	Property	Advanced Extraction T.	Type Value	Comparison Operator	Target Value	operation
Result					Successful	
Response value (JSON)	affected_row_count	N/A		==	1	

6.4.12.15 MongoDBQuery

This system keyword is used for query operations on the MongoDB database. A maximum of 100 records can be queried in the system.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address
Port	Yes	Integer	Database port
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
DB Name	Yes	String	Database instance name
Collection	Yes	String	Collection name
Query	No	String	Query conditions (data in BSON format). By default, this parameter is left blank to query all data in the collection. A maximum of 100 records can be queried in the system.
Limit	No	Integer	- Maximum number of records in the query result. By default, this parameter is left blank to query all data in the collection. A maximum of 100 records can be queried in the system. - If the value is a number out the range of 0 to 100, a maximum of 100 records can be queried in the system.

MongoDBQuery Response

Status: success

Parameter	Type	Description
[<i>Array elements</i>]	Array of row objects	Structure returned by the query result list

Parameter Description

Parameter	Type	Description
field1	String	Database field 1
field2	Integer	Database field 2
...	String	Database field <i>n</i>

MongoDBQuery Response Example

```
[ {
  "_id" : {
    "$oid" : "62465c42907c00003d0076fe"
  },
  "title" : "MongoDB Tutorial",
  "description" : "MongoDB is a NoSQL database.",
  "by" : "test",
  "url" : "",
  "tags" : [ "mongodb", "database", "NoSQL" ],
  "likes" : 100.0
}, {
  "_id" : {
    "$oid" : "62465ce4907c00003d0076ff"
  },
  "title" : "PHP Tutorial",
  "description" : "PHP is a powerful server-side scripting language for creating dynamic interactive sites.",
  "by" : "test",
  "url" : "",
  "tags" : [ "php" ],
  "likes" : 200.0
}, {
  "_id" : {
    "$oid" : "62465ce8907c00003d007700"
  },
  "title" : "Java Tutorial",
  "description" : "Java is a high-level programming language launched by Sun Microsystems in May 1995.",
  "by" : "test",
  "url" : "",
  "tags" : [ "java" ],
  "likes" : 150.0
}]
```

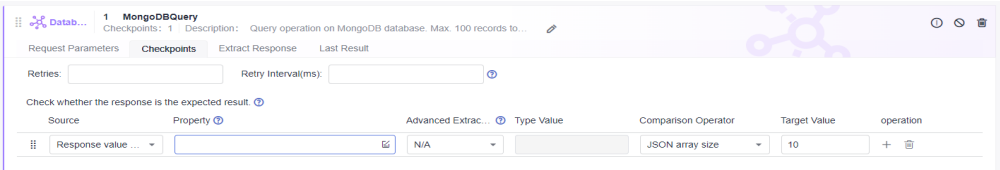
MongoDBQuery Usage Example

- Query the number of documents in the database collection. The system returns the document data list based on the system keyword. You can determine the returned result based on the service.

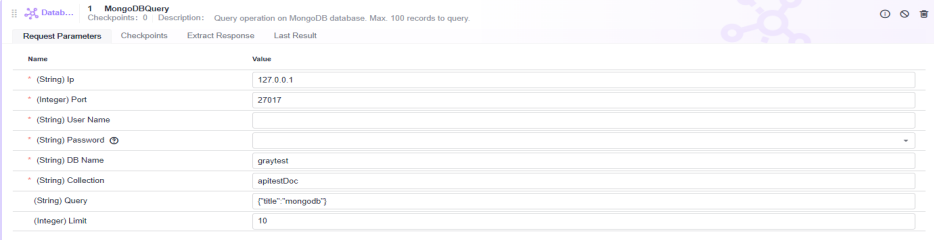
The screenshot shows the 'Request Parameters' tab of the MongoDBQuery configuration. It contains a table with the following parameters:

Name	Value
*(String) Ip	127.0.0.1
*(Integer) Port	27017
*(String) User Name	
*(String) Password	
*(String) DB Name	graytest
*(String) Collection	apitestDoc
(String) Query	
(Integer) Limit	

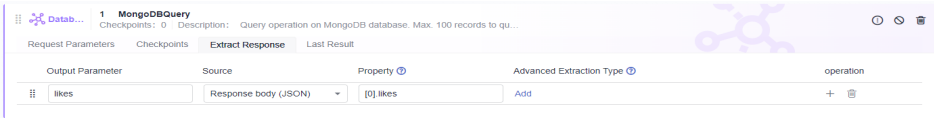
You can determine the result based on the returned list. For example, the number of documents in the collection is 10.



- Query user data and extract data. The queried data list is returned.



Extract or determine the result from response information.



6.4.12.16 MongoDBInsert

This system keyword is used to insert documents into collections in the MongoDB database.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address
Port	Yes	Integer	Database port
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
DB Name	Yes	String	Database instance name
Collection	Yes	String	Collection name
Bson	Yes	String	Inserted data (in BSON format)

MongoDBInsert Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by document insertion

MongoDBInsert Response Example

```
{
  "affected_row_count": 1
}
```

MongoDBInsert Usage Example

Insert data into a collection based on the input test parameter. The system keyword returns the number of inserted records.

Check whether the result is successful through the test checkpoint.

6.4.12.17 MongoDBUpdate

This system keyword is used to update documents in a collection based on input parameters.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address
Port	Yes	Integer	Database port
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
DB Name	Yes	String	Database instance name
Collection	Yes	String	Collection name

Parameter	Mandatory	Type	Description
Query	Yes	String	Updated query conditions (in BSON format)
Update	Yes	String	Document data to be updated (in BSON format)
Multi	No	Boolean	Whether to update all records. false (default): Update only the first record that meets the conditions. true : Update all records that meet the conditions.

MongoDBUpdate Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by document update

MongoDBUpdate Response Example

```
{
  "affected_row_count": 1
}
```

MongoDBUpdate Usage Example

Update documents in the collection based on the entered query data and update data. The system keyword returns the number of updated records.

The screenshot shows the 'Request Parameters' tab of the 'MongoDBUpdate' test case. The parameters are as follows:

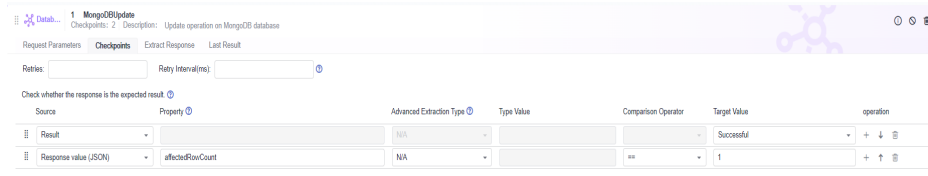
Name	Value
(String) Ip	127.0.0.1
(Integer) Port	27017
(String) User Name	
(String) Password	
(String) DB Name	graytest
(String) Collection	apitestDoc
(String) Query	{"title":"mongodb"}
(String) Update	{"title":"mongodb"}
(Boolean) Multi	false

Update all matched documents when the **multi** parameter is set to **true**.

The screenshot shows the 'Request Parameters' tab of the 'MongoDBUpdate' test case, with the 'Multi' parameter set to 'true'.

Name	Value
(String) Ip	127.0.0.1
(Integer) Port	27017
(String) User Name	
(String) Password	
(String) DB Name	graytest
(String) Collection	apitestDoc
(String) Query	{"title":"mongodb"}
(String) Update	{"title":"mongodb"}
(Boolean) Multi	true

Check whether the result is successful and whether the number of updated records is expected.



6.4.12.18 MongoDBDelete

This system keyword is used to delete documents from a collection based on input parameters.

Parameter	Mandatory	Type	Description
Ip	Yes	String	Database IP address
Port	Yes	Integer	Database port
User Name	Yes	String	Username
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
DB Name	Yes	String	Database instance name
Collection	Yes	String	Collection name
Query	Yes	String	Condition (in BSON format) for deletion. By default, this parameter is left blank to query all data in the collection and delete the data.
JustOne	No	Boolean	Whether to delete a single data record. false : Delete all documents that meet the conditions. true (default) or not configured: Delete only one document that meets the conditions.

MongoDBDelete Response

Status: success

Parameter	Type	Description
affected_row_count	Integer	Number of rows affected by document deletion

MongoDBDelete Response Example

```
{
  "affected_row_count" : 1
}
```

MongoDBDelete Usage Example

Delete documents based on the entered query parameters. The system keyword returns the number of deleted records.

Name	Value
(String) Ip	127.0.0.1
(Integer) Port	27017
(String) User Name	
(String) Password	
(String) DB Name	graytest
(String) Collection	apitestDoc
(String) Query	{"title":"mongodb"}
(Boolean) JustOne	true

If the **JustOne** parameter is set to **false**, all matched documents are deleted.

Name	Value
(String) Ip	127.0.0.1
(Integer) Port	27017
(String) User Name	
(String) Password	
(String) DB Name	graytest
(String) Collection	apitestDoc
(String) Query	{"title":"mongodb"}
(Boolean) JustOne	false

Check whether the result is successful and whether the number of updated records is expected.

Source	Property	Advanced Extraction T.	Type Value	Comparison Operator	Target Value	operation
Result		N/A			Successful	+ ↓
Response value (JSON)	affectedRowCount	N/A		==	1	+ ↑

6.4.12.19 RedisGet

This system keyword is used for Redis character string operations to obtain the value of a specified key.

Parameter	Mandatory	Type	Description
Ip	Yes	String	IP address of the Redis database
Port	Yes	Integer	Redis database port

Parameter	Mandatory	Type	Description
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Key	Yes	String	Specified key name

RedisGet Response

Status: success

Parameter	Type	Description
key	String	Value of the specified key

RedisGet Response Example

```
{
  "test" : "Redis"
}
```

RedisGet Usage Example

- Obtain the value based on the entered key value.

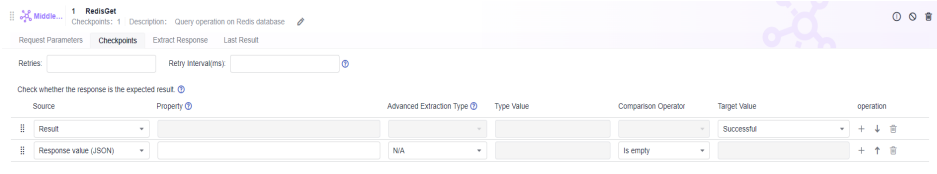
The screenshot shows the 'Request Parameters' tab of the RedisGet configuration. It lists four parameters: (String) Ip with value 127.0.0.1, (Integer) Port with value 6379, (String) Password with a dropdown arrow, and (String) Key with value test.

Check whether the result is successful and expected.

The screenshot shows the 'Checkpoints' tab of the RedisGet configuration. It displays a table for defining checkpoints. The first row is for 'Result' with a comparison operator of 'Successful'. The second row is for 'Response value (JSON)' with a value of 'test', a comparison operator of 'Equals', and a target value of 'Redis'.

If the key does not exist, the returned value is empty. You can determine it at the checkpoint.

The screenshot shows the 'Request Parameters' tab of the RedisGet configuration. It lists four parameters: (String) Ip with value 127.0.0.1, (Integer) Port with value 6379, (String) Password with a dropdown arrow, and (String) Key with value taskid.



6.4.12.20 RedisSet

This system keyword is used for Redis character string operations. You can configure the expiration time and value based on a specified key.

Parameter	Mandatory	Type	Description
Ip	Yes	String	IP address of the Redis database
Port	Yes	Integer	Redis database port
Password	Yes	String	Password (The login password is personal information and must be defined as sensitive in the environment parameters.)
Key	Yes	String	Specified key name
Value	Yes	String	Specified value
Expire	No	Long	Expiration time, in seconds. If the default value -1 is used, the password never expires. If a negative integer of the Long type is used, the password never expires.

RedisSet Response

Status: success

Parameter	Type	Description
result	String	Result returned by the configured value corresponding to the key. If the setting is successful, OK is returned. If the setting fails, an error message is returned.

RedisSet Response Example

```
{  
  "result" : "OK"  
}
```

RedisSet Usage Example

- Set the value based on the entered key value. The default value is used, indicating that the password never expires.

The screenshot shows the 'Request Parameters' tab for a test case named 'RedisSet'. The parameters are as follows:

Name	Value
(String) Ip	127.0.0.1
(Integer) Port	6379
(String) Password	
(String) Key	test:cloud
(String) Value	redis
(Long) Expire	-1

Check whether the setting is successful.

The screenshot shows the 'Checkpoints' tab for the 'RedisSet' test case. It includes a 'Retries' field set to 1 and a 'Check whether the response is the expected result' section. The table below shows the expected result configuration:

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value	operation
Result					Successful	+ ↓
Response value (JSON)	result	N/A		Equals	OK	+ ↑

- Set the corresponding value based on the entered key value. Set the expiration time to 600 seconds.

The screenshot shows the 'Request Parameters' tab for the 'RedisSet' test case. The parameters are as follows:

Name	Value
(String) Ip	127.0.0.1
(Integer) Port	6379
(String) Password	
(String) Key	test:cloud
(String) Value	redis
(Long) Expire	600

Check whether the setting is successful.

The screenshot shows the 'Checkpoints' tab for the 'RedisSet' test case. It includes a 'Retries' field set to 1 and a 'Check whether the response is the expected result' section. The table below shows the expected result configuration:

Source	Property	Advanced Extraction T...	Type Value	Comparison Operator	Target Value	operation
Result					Successful	+ ↓
Response value (JSON)	result	N/A		Equals	OK	+ ↑

6.4.12.21 OBSWrite

This system keyword is used for OBS string operations, that is, to set the value based on a specified key.

Parameter	Mandatory	Type	Description
Access Key ID	Yes	String	OBS AK (This personal information must be defined as sensitive in the environment parameters.)
Secret Access ID	Yes	String	OBS SK (This personal information must be defined as sensitive in the environment parameters.)
Rest Endpoint	Yes	String	OBS endpoint
Bucket Name	Yes	String	OBS bucket name
Key	Yes	String	OBS file path
Value	Yes	String	OBS file content

OBSWrite Response

Status: success

Parameter	Type	Description
result	String	If the setting is successful, ok is returned.
key	String	OBS file path

Status: failed

Parameter	Type	Description
result	String	If the setting is successful, fail is returned.

Parameter	Type	Description
errorMessage	String	Failure cause

OBSWrite Response Example

```
{
  "result" : "ok",
  "key" : "/key"
}
```

OBSWrite Usage Example

Write the test data to the **key1/key2** path.

Check whether the setting is successful.

6.4.12.22 OBSDelete

This system keyword is used for deleting OBS files based on a specified key.

Parameter	Mandatory	Type	Description
Access Key ID	Yes	String	OBS AK (This personal information must be defined as sensitive in the environment parameters.)
Secret Access ID	Yes	String	OBS SK (This personal information must be defined as sensitive in the environment parameters.)

Parameter	Mandatory	Type	Description
Rest Endpoint	Yes	String	OBS endpoint
Bucket Name	Yes	String	OBS bucket name
Key	Yes	String	OBS file path

OBSDelete Response

Status: success

Parameter	Type	Description
result	String	If the setting is successful, ok is returned.
key	String	OBS file path

Status: failed

Parameter	Type	Description
result	String	If the setting is successful, fail is returned.
errorMessage	String	Failure cause

OBSDelete Response Example

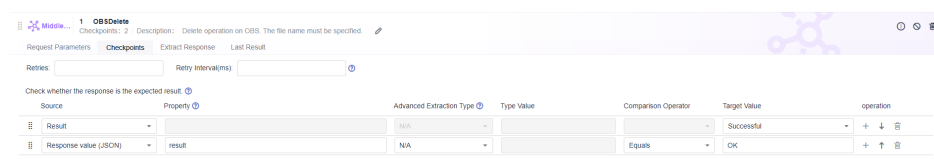
```
{
  "result" : "ok",
  "key" : "/key"
}
```

OBSDelete Usage Example

Delete data based on the **key1/key2** path.



Check whether the setting is successful.



6.4.12.23 OBSQuery

This system keyword is used for querying OBS content, that is, to obtain the file content based on a specified key.

Parameter	Mandatory	Type	Description
Access Key ID	Yes	String	OBS AK (This personal information must be defined as sensitive in the environment parameters.)
Secret Access ID	Yes	String	OBS SK (This personal information must be defined as sensitive in the environment parameters.)
Rest Endpoint	Yes	String	OBS endpoint
Bucket Name	Yes	String	OBS bucket name
Key	Yes	String	OBS file path

OBSQuery Response

Status: success

The file content is directly returned, for example, "Test data".

```
"Test data"
```

Status: failed

Parameter	Type	Description
result	String	If the setting is successful, fail is returned.
errorMessage	String	Failure cause

OBSQuery Usage Example

Query data based on the **key1/key2** path.

1 OBSQuery

Checkpoints: 0

Description: Query operation on OBS. The file name must be specified.

Request Parameters

Checkpoints

Extract Response

Last Result

Name	Value
(String) Access Key ID	\$(AK)
(String) Secret Access ID	\$(SK)
(String) Rest Endpoint	obs- XXXXXXXXXXXX
(String) Bucket Name	XXXXXXXXXXXX
(String) Key	key1.key2

Check whether the data is obtained successfully.

1 OBSQuery

Checkpoints: 0

Description: Query operation on OBS. The file name must be specified.

Request Parameters

Checkpoints

Extract Response

Last Result

Retries: Retry Interval(ms):

Check whether the response is the expected result.

Source	Path	Advanced Extraction Type	Type Value	Comparison Operator	Target Value	operation
Response value (JSON)	Enter a value.	N/A		Equals	Testdata	+

6.4.12.24 KafkaProducer

Introduction to KafkaProducer

This system keyword can be used to test Kafka producers.

Parameter	Mandatory	Type	Default Value	Description
Broker	Yes	String	127.0.0.1:9093	IP address of a Kafka instance
Topic	Yes	String	-	Topic of a Kafka message
Message	Yes	String	-	Body of each Kafka message
SASL Username	No	String	-	Kafka SASL username
SASL Password	No	String	-	Kafka SASL password
Truststore	No	File	-	Kafka client certificate
Truststore Password	No	String	-	Password of the Kafka client certificate

Kafka Authentication Description

- Authentication mode: If you configure **SASL Username**, **SASL Password**, **Truststore**, and **Truststore Password** in the **KafkaProducer** keyword, and the

Kafka uses the SASL_SSL authentication protocol for encrypted data transmission, then client authentication is required.

- No authentication mode: If you do not configure **SASL Username**, **SASL Password**, **Truststore**, and **Truststore Password** in the **KafkaProducer** keyword, and the Kafka uses the default SASL_PLAINTEXT protocol, then no encryption or authentication mechanism will be used. In this mode, the communication between the client and server is in plaintext, and security cannot be guaranteed.

KafkaProducer Response

Status: success

Parameter	Type	Description
Body	String	Value returned by the KafkaProducer API

KafkaProducer Response Example

```
{
  "result" : "send message to kafka success.",
  "status" : "ok"
}
```

Default Checkpoint

Name	Expected Value
Result	Success

KafkaProducer Usage Example

Configure parameters to send a Kafka message.



The screenshot shows the 'Request Parameters' tab for the 'KafkaProducer' keyword. The form includes the following fields:

Name	Value
* (String) Broker	127.0.0.1:9093
* (String) Topic	apitest-topic
* (String) Message	kafka message 1
(String) SASL Username	test
(String) SASL Password	
(File) Truststore	--select--
(String) Truststore Password	

6.4.12.25 KafkaConsumer

Introduction to KafkaConsumer

This system keyword can be used to test Kafka consumers.

Parameter	Mandatory	Type	Default Value	Description
Broker	Yes	String	127.0.0.1:9093	IP address of a Kafka instance
Topic	Yes	String	-	Topic of a Kafka message
Consumer Group	Yes	String	-	Kafka message consumer group
SASL Username	No	String	-	Kafka SASL username
SASL Password	No	String	-	Kafka SASL password
Truststore	No	File	-	Kafka client certificate
Truststore Password	No	String	-	Password of the Kafka client certificate

Kafka Authentication Description

- Authentication mode: If you configure **SASL Username**, **SASL Password**, **Truststore**, and **Truststore Password** in the **KafkaProducer** keyword, and the Kafka uses the SASL_SSL authentication protocol for encrypted data transmission, then client authentication is required.
- No authentication mode: If you do not configure **SASL Username**, **SASL Password**, **Truststore**, and **Truststore Password** in the **KafkaProducer** keyword, and the Kafka uses the default SASL_PLAINTEXT protocol, then no encryption or authentication mechanism is used. In this mode, the communication between the client and server is in plaintext, and security cannot be guaranteed.

KafkaConsumer Response

Status: success

Parameter	Type	Description
Body	String	Value returned by the KafkaConsumer API

KafkaConsumer Response Example

```
{
  "message" : [{
    "offset" : 102130,
    "value" : "kafka message 1"
  }, {
    "offset" : 102131,
    "value" : "kafka message 2"
  }]
}
```

Default Checkpoint

Name	Expected Value
Result	Success

KafkaConsumer Usage Example

Configure parameters to receive a Kafka message.

Name	Value
(String) Broker	127.0.0.1:9093
(String) Topic	apitest-topic
(String) Consumer Group	apitest-consumer-group
(String) SASL Username	test
(String) SASL Password	
(File) Truststore	--select--
(String) Truststore Password	

6.4.12.26 TCP

TCP

This system keyword is used to test the TCP basic protocol. Ensure that the user IP address is accessible from the public network and the security group policy of the corresponding port is enabled to permit access.

Parameter	Mandatory	Type	Default Value	Description
Host	Yes	String	-	TCP service address
Port	Yes	Integer	-	TCP service port
Connect Timeout	Yes	Integer	-	TCP service connection timeout Unit: millisecond

Parameter	Mandatory	Type	Default Value	Description
Read Timeout	Yes	Long	-	Message reading timeout Unit: millisecond
Check End Type	Yes	Enum	CheckEndStr	Message end flag: CheckEndLength CheckEndStr
Body Type	Yes	Enum	CharSequence	Message body type: CharSequence HexadecimalCodeStream

TCP Response

Status: success

Parameter	Type	Description
Body	String	TCP API return

TCP Response Example

```
{  
  "This is a test Message."  
}
```

Default Checkpoint

Name	Expected Value
Result	Success

TCP Usage Example

Configure parameters to access the TCP service.

The screenshot shows the 'Request Parameters' configuration for a TCP test case. The parameters are as follows:

Name	Value
(String) Host	127.0.0.1
(Integer) Port	8080
(Integer) Connect Timeout	500
(Long) Read Timeout	500
(Enum) Check End Type	CheckEndStr (selected)
(Enum) Body Type	CharSequence (selected)
Message Body	This is a test message!

6.4.12.27 UDP

UDP

This system keyword is used to test the UDP basic protocol. Ensure that the user IP address is accessible from the public network and the security group policy of the corresponding port is enabled to permit access.

Parameter	Mandatory	Type	Default Value	Description
Host	Yes	String	-	UDP service address
Port	Yes	Integer	-	UDP service port
Check End Type	Yes	Enum	CheckEndStr	Message end flag: CheckEndLength CheckEndStr
Body Type	Yes	Enum	CharSequence	Message body type: CharSequence HexadecimalCodeStream

UDP Response

Status: success

Parameter	Type	Description
Body	String	UDP API return

UDP Response Example

```
{  
  "This is a test Message."  
}
```

Default Checkpoint

Name	Expected Value
Result	Success

UDP Usage Example

- Configure parameters to access the UDP service.

The screenshot shows the 'UDP' configuration window. The 'Request Parameters' tab is selected. It contains the following fields:

- Name:** (String) Host, Value: 127.0.0.1
- Name:** (Integer) Port, Value: 8080
- Name:** (Enum) Check End Type, with radio buttons for ☒ CheckEndLength and ☐ CheckEndStr, and a value field containing 30.
- Name:** (Enum) Body Type, with a dropdown menu set to 'CharSequence' and a text field containing 'This is a test Message!'.

6.4.12.28 WSConnect

This system keyword is used for connecting a WebSocket client to a server.

Parameter	Mandatory	Type	Description
Request Uri	Yes	String	WebSocket service address
Response Timeout	Yes	Long	Response timeout period
Header	Yes	String	Request header
Connect Timeout	Yes	Integer	Connection timeout period

WSConnect Response

Status: success

Parameter	Type	Description
Body	String	Value returned by the WSConnect API

WSConnect Response Example

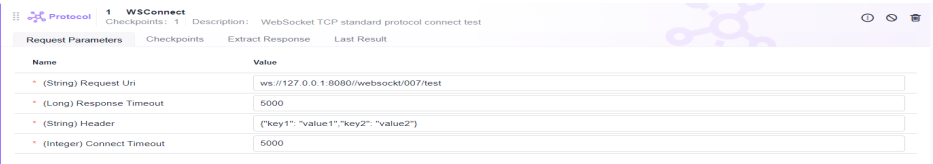
```
{
  "Connect to *** at port *** in time 5000 successfully."
}
```

Default Checkpoint

Name	Expected Value
Result	Success

WSConnect Usage Example

Configure parameters to access the WebSocker service.



6.4.12.29 WSRequest

This system keyword is applicable to the scenario where a WebSocket client requests a server operation.

Parameter	Mandatory	Type	Description
Request Uri	Yes	String	WebSocket service address
Response Timeout	Yes	Long	Response timeout period
Header	Yes	String	Request header
Request Type	Yes	String	Request data type
Request Body	Yes	String	Request data
Response Type	Yes	String	Response data type

WSRequest Response

Status: success

Parameter	Type	Description
Body	String	Value returned by the WSRequest API

WSRequest Response Example

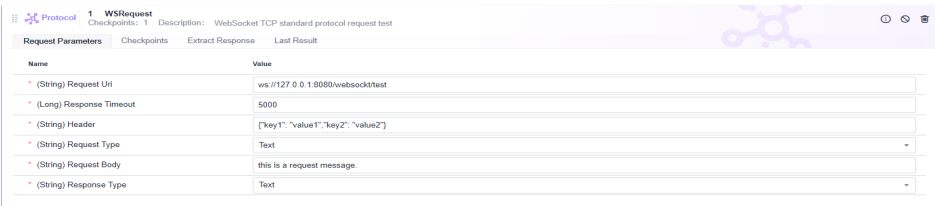
```
{
  "Write data to *** at port *** successfully. response: this is a response message."
}
```

Default Checkpoint

Name	Expected Value
Result	Success

WSRequest Usage Example

Configure parameters to access the WebSocker service.



6.4.12.30 WSWriteOnly

This system keyword is used to access a WebSocket server for write-only operations.

Parameter	Mandatory	Type	Description
Request Uri	Yes	String	WebSocket service address
Response Timeout	Yes	Long	Response timeout period
Header	Yes	String	Request header
Request Type	Yes	String	Request data type
Request Body	Yes	String	Request data

WSWriteOnly Response

Status: success

Parameter	Type	Description
Body	String	Value returned by the WSWriteOnly API

WSWriteOnly Response Example

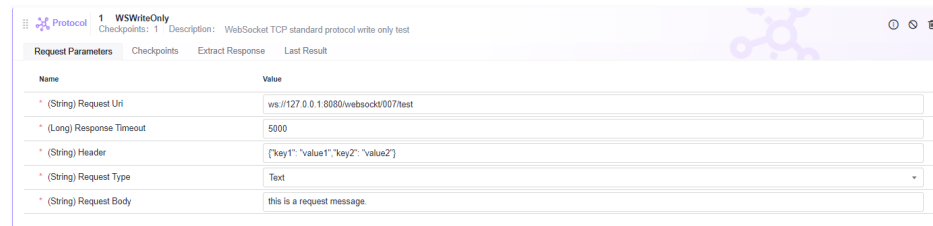
```
{
  "Write data to *** at port *** successfully. response: this is a response message."
}
```

Default Checkpoint

Name	Expected Value
Result	Success

WSWriteOnly Usage Example

Configure parameters to access the WebSocker service.



6.4.12.31 WSReadOnly

This system keyword is used to access a WebSocket server for read-only operations.

Parameter	Mandatory	Type	Description
Request Uri	Yes	String	WebSocket service address
Response Timeout	Yes	Long	Response timeout period
Header	Yes	String	Request header
Response Type	Yes	String	Response data type

WSReadOnly Response

Status: success

Parameter	Type	Description
Body	String	Value returned by the WSReadOnly API

WSReadOnly Response Example

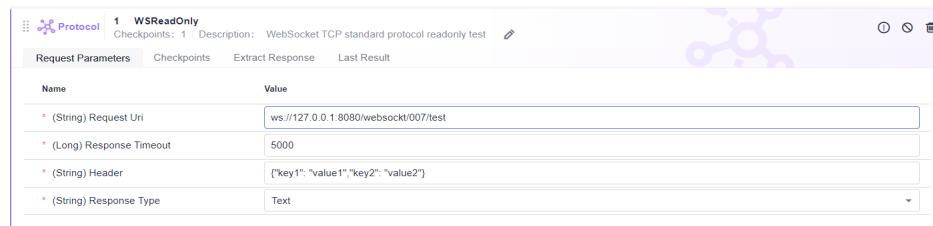
```
{
  "Read data from*** at port *** successfully. response: this is a response message."
}
```

Default Checkpoint

Name	Expected Value
Result	Success

WSReadOnly Usage Example

Configure parameters to access the WebSocker service.



6.4.12.32 WSDisConnect

This system keyword is used to access a WebSocket server for disconnection.

Parameter	Mandatory	Type	Description
Request Uri	Yes	String	WebSocket service address
Response Timeout	Yes	Long	Response timeout period
Header	Yes	String	Request header
Status Code	Yes	String	Response data type
Message	Yes	String	Disconnection message

WSDisConnect Response

Status: success

Parameter	Type	Description
Body	String	Value returned by the WSDisConnect API

WSDisConnect Response Example

```
{
  "Disconnect to *** at port *** in time 5000 successfully."
}
```

Default Checkpoint

Name	Expected Value
Result	Success

WSDisConnect Usage Example

Configure parameters to access the WebSocker service.



Name	Value
(String) Request Uri	ws://127.0.0.1:8080/websocket/007/test
(Long) Response Timeout	5000
(String) Header	{"key1": "value1", "key2": "value2"}
(Integer) Status Code	1000
(String) Message	close connect

6.4.12.33 DubboClient

Introduction to Dubbo

Apache Dubbo is a microservice development framework that provides RPC communication and microservice governance. Microservices developed using Dubbo can remotely discover and communicate with each other. In addition, Dubbo provides rich service governance capabilities to meet service governance requirements such as service discovery, load balancing, and traffic scheduling.

Introduction to DubboClient

This system keyword is used to test the Dubbo protocol. Ensure that the Dubbo service is accessible from the public network and the security group policy of the corresponding port is enabled to permit access.

Parameter	Mandatory	Type	Default Value	Description
Dubbo Server IP Address	Yes	String	-	IP address of the Dubbo service
Dubbo Server Port	Yes	Integer	-	Dubbo service port

Parameter	Mandatory	Type	Default Value	Description
Dubbo operation instruction	Yes	String	LS	Dubbo operation command. The options are LS and INVOKE .
Dubbo Registration API	Yes	String	-	Name of the API registered with Dubbo. It is a fully qualified class name.
Dubbo Registration API	Yes	String	-	Method signature corresponding to the API registered with Dubbo

DubboClient Response

- Status: success

Parameter	Type	Description
Body	String	Value returned by the Dubbo service

Default Checkpoint

Name	Expected Value
Result	Success

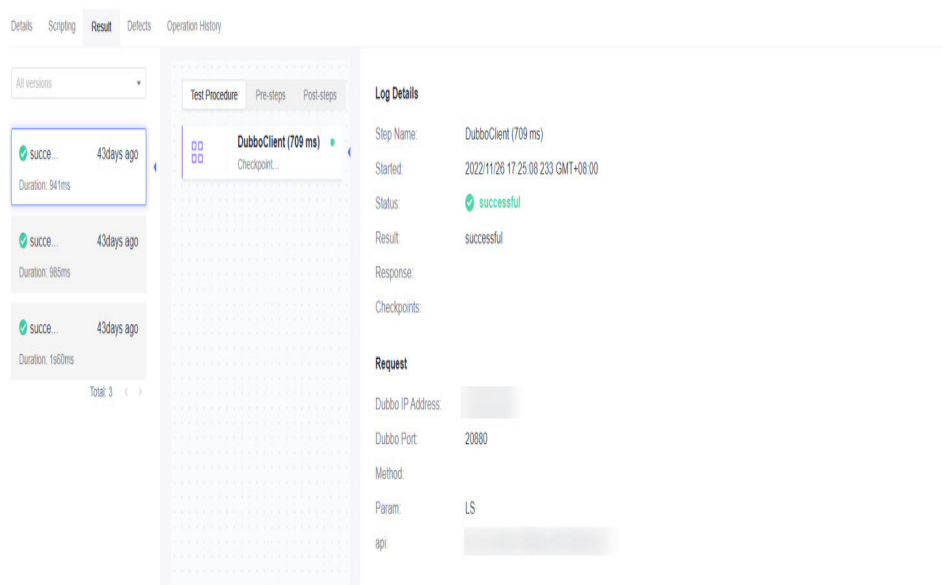
DubboClient Usage Example: Obtaining All Registered APIs of the Server

Parameter example:

The screenshot shows the 'Request Parameters' tab of the DubboClient configuration. It lists five parameters with their types and values:

Name	Value
* (String) Dubbo Server IP Address	
* (Integer) Dubbo Server Port	20880
* (String) Dubbo Operation Instruction	LS
* (String) Dubbo Registration API	
* (String) Dubbo Registration Method	

Response example:

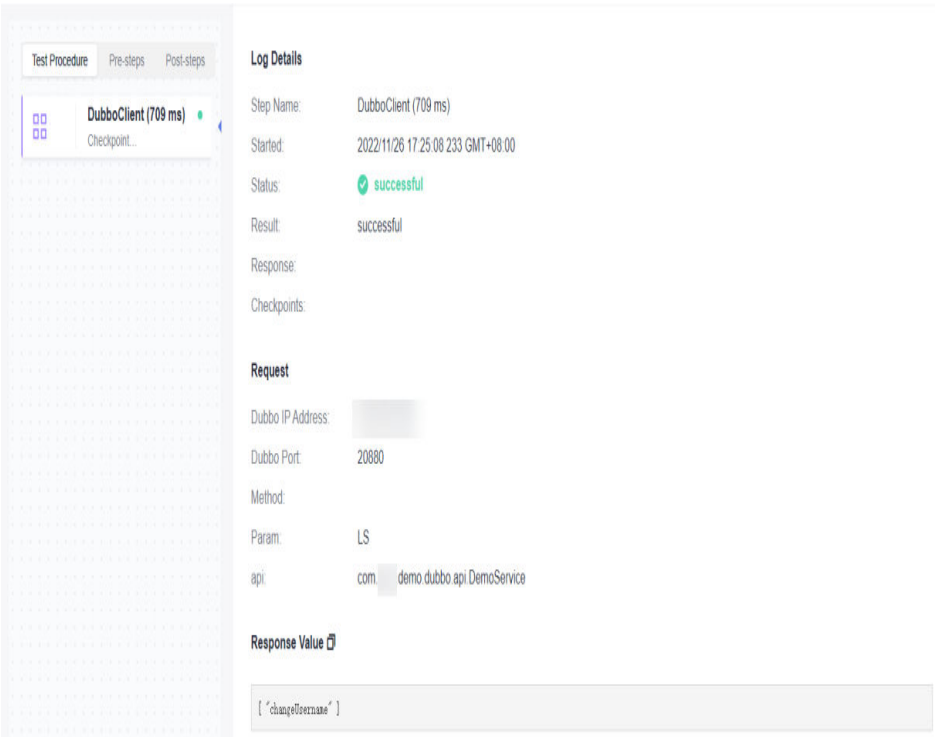


DubboClient Usage Example: Obtaining the Specified Service Registration API of the Server

Parameter example:



Response example:

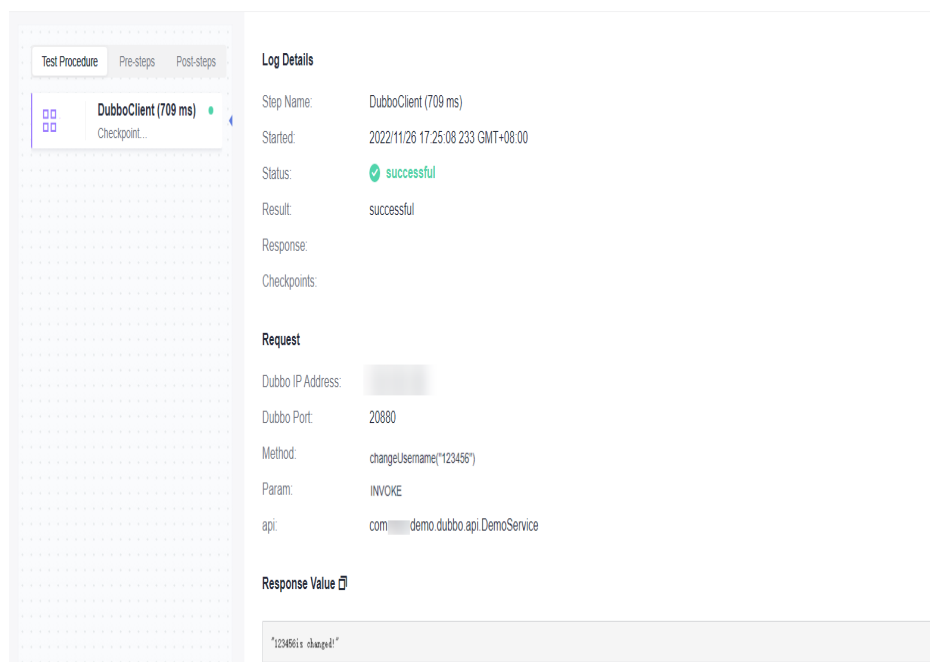


DubboClient Usage Example: Calling a Specified Dubbo API

Parameter example:



Response example:



6.5 Executing a Test Case

Execute test cases and record the results.

Notes and Constraints

Up to five API automation test cases can be executed in a batch each time.

Common Execution

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** Click the tab that corresponds to the execution type of the test case. For example, click the **Manual Test** tab for a manual test case, or the **Auto API Test** for an automated API test case. Click  in the **Operation** column of the test case. The execution page is displayed.

If multiple API automation test cases need to be executed at the same time, select them and click **Execute**.

NOTE

Redirection in API testing refers to the process of forwarding a request for a URL or web page to another URL or web page. The server forwards the user's access request for a URL to another URL based on certain rules. The user will then see the page of the latter URL.

This redirection function can be enabled in **Function Switch** on the **Test Settings** page. After this function is enabled, if the request code is **302** during execution of API automation test cases, the system will automatically forward the request to the destination redirect address in the URL.

Step 4 For manual test cases, set the actual test step result and case result, and click **Save** to return to the case list.

Click **Add** to upload local attachments related to the test activity to the execution history. Up to five attachments can be uploaded, max. 10 MB each.

Step 5 After the execution is complete, check the execution result in the **Result** column of the test case list.

Step 6 Click the test case name, click the execution history tab, and view the execution history of the test case.

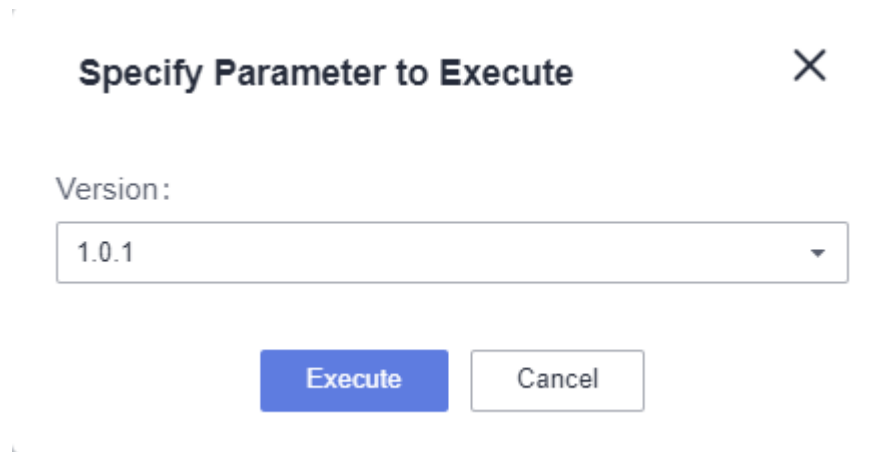
----End

Executing a Manual Test Case with Parameters

Test cases can be executed with parameters.

Step 1 In the test case list, click **...** in the **Operation** column and select **Specify Parameter to Execute**.

Step 2 In the displayed dialog box, enter the version number and click **Execute**. The execution page is displayed.



Step 3 Set the case execution result, enter the actual result, and click **Save**.

Step 4 Click the corresponding case and click the execution history tab to query the execution result.

----End

Execution of an Automated API Test Case with a Dataset

Step 1 [Import an automated API test case dataset](#).

Step 2 In the navigation pane, choose **Testing > Testing Case**.

Step 3 Click the **Auto API Test** tab, locate the test case to be executed, and click  in the **Operation** column to start automatic execution.

Step 4 After the execution is complete, check the execution result in the **Result** column of the test case list.

Click the test case name and the execution history tab, and view the execution history of the test case.

Step 5 View the number of failed rounds and the total number of dataset rounds.

Step 6 Click  on the card to view the execution status of each round of a dataset.

Click the drop-down list box to filter data by **Successful** or **Failed**.

Step 7 Click a round. The detailed log information about the round is displayed.

----End

6.6 Managing Test Cases

You can import test cases from the local PC to the test case library in CodeArts TestPlan, and export test cases from the test case library. You can also add test cases in batches, manage test cases through the test feature directory, associate test cases with requirements, comment on test cases, filter test cases, customize the columns to be displayed in the test case list, and set test case fields.

Notes and Constraints

- When importing test cases from a file, make sure that the file size is smaller than or equal to 5 MB, and that the number of test cases in the version will not exceed the package limit.
- A maximum of 500 API automation test cases can be imported using an Excel file.
- A test case can be associated with up to 100 requirements.
- API automation test cases can be imported either by Postman and Swagger files. The file types can be:

Postman: Postman Collection v2.1 standard, Postman Collection JSON file

Swagger: Swagger 2.0 and 3.0 standards, YAML file

Importing Manual Test Cases

Import manual test cases using an Excel file or from other projects.

Importing a Manual Test Case from Excel

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Click the **Manual Test** tab, click **Import** on the right of the page, and choose **Import from File** from the drop-down list.

Alternatively, click the **All Cases** tab, click **Import** on the right of the page, and choose **Import from File** from the drop-down list.

Step 3 Decide whether to allow the uploaded cases to overwrite the existing cases with the same IDs, and select the corresponding option.

YES: If the uploaded cases have the same IDs with the existing cases, the existing ones will be overwritten.

NO: All cases will be uploaded to the case list.

Step 4 In the displayed dialog box, select **Whether to create a directory**.

The directory corresponds to the test feature directory on the page. Directory structure: level-1 subdirectory, level-2 subdirectory, level-3 subdirectory, and level-4 subdirectory. The level-1 subdirectory is the first directory under the **Test Features** directory.

Step 5 In the displayed dialog box, click **Download Template**.

Enter the test case information based on the format requirements in the template, return to the **Testing Case** page, upload the created test case file, and click **OK**.

When you use the Excel file for [exporting test cases](#) to import the test cases again, these fields will not be imported: result, result remarks, directory ID, creator, creation time, version, executor, execution time, execution duration, execution times, review status, directory level, and requirement name.

If the data in the Excel file is invalid, an error message will be displayed. Click **Download** to view the error cause, modify the data, and import the file again.

----End

Adding Manual Test Cases from Another Project

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Click the **Manual Test** tab, click **Import** on the right of the page, and choose **Import from Project** from the drop-down list.

Step 3 In the displayed dialog box, select the source project.

Step 4 Select a source version.

Step 5 Select or deselect the overwriting rule for the test cases with duplicate IDs.

- If the rule is selected, the uploaded cases will overwrite the existing cases with the same IDs.
- If the rule is deselected, the uploaded cases will not be imported if they have the same IDs as the existing cases.

Step 6 Select the original test feature or test requirement to which the test cases belong. You can find them by using the search box on the left.

Step 7 Search for or filter the required cases by using the search box above the test case list.

Step 8 Select the cases and click **Target Folder**. In the displayed dialog box, select the feature or requirement to which the cases are imported, and click **OK**.

If you select **Synchronize cases only**, only the cases in the original directory are imported to the target directory.

Step 9 Click **OK**.

If no target directory is specified, selected cases are saved under **Test Features** while remaining in their original directory.

----End

Importing API Automation Test Cases

CodeArts TestPlan allows you to import API automation test cases in batches from Postman, Swagger, or Excel files.

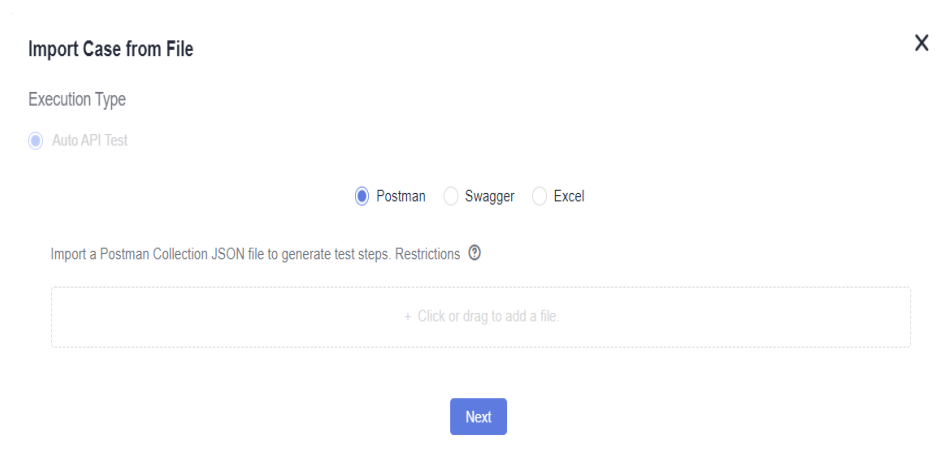
Importing Postman or Swagger Files to Generate API Automation Test Cases

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Click the **Auto API Test** tab and choose **Import > Import from File** on the right of the page. The **Import Case from File** window is displayed.

Step 3 Select **Postman** or **Swagger**.

Drag a file from the local PC to the window, or click **Click or drag to add a file** and select a file from the local PC. Click **Next**.



Step 4 In the displayed list, select the items that you want to create test cases for in the specified order, and click **Save**.

----End

Generating API Automation Test Cases by Importing an Excel File

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Click the **Auto API Test** tab and choose **Import > Import from File** on the right of the page.

Alternatively, click the **All Cases** tab, click **Import** on the right of the page, and choose **Import from File** from the drop-down list. In the displayed dialog box, set **Execution Type** to **Auto API Test**.

Step 3 Select **Excel** and click **Download Template**.

Import Case from File

X

Execution Type

☒ Auto API Test

☐ Postman

☐ Swagger

☒ Excel

Download Template

Follow comments to fill in the template. A maximum of 500 cases can be imported at a time.

+ Click or drag to add a file.

Import

Cancel

Step 4 Open the Excel template on the local PC and edit the test case information based on the comments in the template headers. The columns marked with asterisks (*) are required.

The following table shows the template fields.

Field	Description
Case Name (mandatory)	The value can contain 1 to 128 characters. Only letters, digits, and special characters (-_/*&`'^~;:(){}=+,x...—!@#\$.%.[<>?-"') are supported.
Case Description	Max. 500 characters.
Request Type (mandatory)	Only GET, POST, PUT, and DELETE are supported.
Request Header Parameter	Format: key=value . If there are multiple parameters, separate them with & , that is, key=value&key1=value1 .
Request Address (mandatory)	The request protocol can be HTTP or HTTPS. The format is https://ip:port/pathParam?query=1 .
Environment Group	Environment parameter group.
IP Variable Name	Generates the variable name in the corresponding Environment Group , extracts the content of the Request Address , and generates the corresponding global variable.
Request Body Type	The value can be raw , json , or formdata , which corresponds to the text, JSON request body, or form parameter format on the page, respectively. If this parameter is not set, the JSON format is used by default.

Field	Description
Request Body	If the request body type is formdata , the request body format is key=value . If there are multiple parameters, separate them with & , that is, key=value&key2=value2 . When cases are imported using an Excel file, formdata does not support the request body in file format.
Checkpoint Matching Mode	Supports exact match and fuzzy match. Exact match indicates Equals , and Fuzzy match indicates Contains .
Expected Checkpoint Value	Target value of the checkpoint.

Step 5 Save the edited Excel file and drag it from the local PC to the **Import Case from File** window, or click **Click or drag to add a file.** and select a file from the local PC. Click **Next**.

Step 6 View the import result.

- Import successful: New test cases are displayed in the list. The number of new test cases is the same as the number of rows in the Excel file.
- Import failed: A failure message is displayed in the upper right corner.
Download the error list from the **Import Case from File** window. Modify the Excel file based on the error causes, and import it again.

----End

Exporting Test Cases

Step 1 Click an execution type tab, click **More** on the right and choose **Export** from the drop-down list.

Step 2 In the displayed dialog box, select the case export scope. You can select **Export All**.

Select **Partial Export**, set **Start position** to the case to be exported to the first row of the table and **End position** to the case to be exported as the last row of the table, and click **OK**.

Step 3 Before exporting, select the fields to be displayed in the local Excel file. (The data selected last time is displayed by default when you export the file next time.)

Step 4 Click **Export**.

Step 5 Open the exported Excel file on the local PC and check the exported test case. (You will see the IDs of bugs and requirements associated with the exported test cases. Multiple IDs are separated by commas.)

----End

Adding Test Cases in Batches from the Test Case Library

Add manual and automated API test cases from the test case library to test plans in batches.

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** Click **Test case library** in the upper left corner of the page and select the target test plan from the drop-down list.
- Step 4** Click the **Manual Test** or **Auto API Test** tab, click **Import** in the right area, and select **Add Existing Cases** from the drop-down list.
- Step 5** In the displayed dialog box, select a test case and click **OK**.
- Test cases that already exist in the test plan cannot be added again.
 - All test cases related to requirements in the test plan can be added.

----End

Reviewing Test Cases Online

Review the created test cases.

Creating a Review

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** Click the tab of the corresponding test type, click **...** on the right of the case to be reviewed, and click **Create Review**.
- Step 4** In the displayed dialog box, configure the following information and click **Confirm**.

Configuration Item	Description
Name	By default, the name of a new review is the same as the test case name.
Test Case Modification Time	By default, the time is set to the current system date.

Config uration Item	Description
Review Auto- Close	You can enable or disable the automatic closure function for the review. • Yes: The review will be automatically closed after it is created. • No: The test case will be manually reviewed and closed by the person to whom the review is assigned to.
Expe cted Closure Time	If you select No for Review Auto-Close , you can select the expected closure time.
Review Comme nt	Enter review information containing a maximum of 1,000 characters.
Assigne d To	If you select No for Review Auto-Close , you can select a person to whom the review is assigned to close.

----End

Batch Review

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 In the test case list, select the test cases to be reviewed in batches.

Step 3 Click **Review**.

Step 4 In the displayed dialog box, configure the following information and click **Confirm**.

- **Review Auto-Close:** If this parameter is set to **Yes**, the review status is automatically set to **Closed**. If this parameter is set to **No**, the test case will be manually reviewed and closed by the person to whom the review is assigned.
- **Expected Closure Time:** If you select **No** for **Review Auto-Close**, you can select the expected closure time.
- **Review Comment:** Enter review information containing a maximum of 1000 characters.
- **Assigned to:** If you select **No** for **Review Auto-Close**, you can select a person to whom the review is assigned for closure.

----End

Checking Review Records

Step 1 Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Case**.

Step 3 Click **Review Record**. The created review record is displayed on the page.

Step 4 Click the search bar on the review record page and select a filter field.

Step 5 Enter a keyword to search for the corresponding review record.

Step 6 Delete, edit, or close reviews that have not been closed.

Review Record

Number of cases :3 Total number of reviews :11 Total closed-loop tree :9 Number of cases without review :1 Review is not closed :2 Reviewed and closed :0

Click here to choose a filter condition.

☐	Name	ID	Plan	Review ...	Reviewer	Review Time	Closer	Actual C...	Clos...	Clos...	Expected	Operation
☐	test	11707	web...	OK	devclo...	May 13, 2024...	lwx	devclo...	CL...	--	May 21, 2024	
☐	test	11707	web...	6	devclo...	May 13, 2024...	lwx	--	OPEN	--	May 13, 2024	

- To delete an unclosed review, click in the **Operation** column and click **OK**.
- To edit an unclosed review record, click in the **Operation**. In the displayed dialog box, edit the review.
- To close an unclosed review, click in the **Operation** column. In the **Close Review** dialog box, close the review.

----End

Associating Test Cases with Project Requirements

CodeArts TestPlan supports association between test cases and requirements.

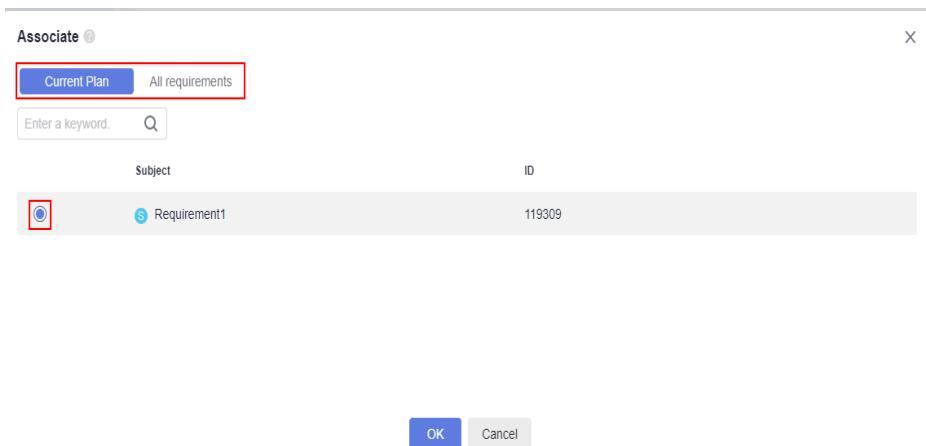
Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Associate test cases with requirements in one of the following ways:

- Click the required test type tab. In the case list, click ******* in the row of the target case and select **Associated With Requirement**.
- Click the **All** tab. In the **Operation** column of the target test case, click to associate the case with a requirement.
- Go to the test case details page, click the **Requirements** tab, and click **Associated With Requirements**.
- To associate multiple test cases with one requirement, select the required test cases in the list and click **Associate Requirement** on the toolbar below the list.

Step 3 In the displayed dialog box, select the requirement to be associated on the **Current Plan** or **All requirements** tab page, and click **OK**.

When you associate the target requirement in **All requirements**, if the requirement has not been added to the current test plan, select **Add it synchronously**. or **Only the requirement is associated**. in the dialog box.



----End

Adding Test Cases from Associated Requirements

Prerequisites

The requirements in the test plan have been associated with test cases in the test case library.

Managing Test Cases by Requirement

- Step 1** On the **Testing Case** page, click the **Test Requirements** tab on the left. By default, the requirements that have been associated are stored in the **Test Requirements** directory.
- Step 2** Click a requirement to view all cases associated with it.
- Step 3** On the right of the requirement name, click  and choose to view the requirement details or create a test case for the requirement.

----End

Adding Test Cases from Requirements Associated with a Test Plan

- Step 1** Click **Test case library** in the upper left corner of the page and select the target test plan from the drop-down list.
- Step 2** Click the **Manual Test** or **Auto API Test** tab, click **Import** in the right area, and select **Add Existing Cases** from the drop-down list.
- Step 3** In the displayed dialog box, select **Select all test cases related to the requirements in this test plan..** The test cases associated with requirements in the test plan are automatically selected.
- Step 4** Click **OK**.

----End

Requirement Change Notification

If a requirement associated to a test case is changed in CodeArts Req, a red dot will show up next to the requirement name on the **Testing Case** page. This signals that you should update the test cases for that requirement or add new ones.

Test Cases and Bugs

When a test case fails to be executed, the test case is usually associated with a bug. You can create a bug or associate the test case with an existing bug.

The following uses manual test cases as an example.

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Select a test case to be associated with a bug. Create or associate a bug in one of the following ways:

- Click  in the **Operation** column to associate an existing bug in the current project.
- Click  in the **Operation** column, and select **Create and Associate Defects** to create a bug as prompted.
- Click a test case to associate it with a bug.
Click the test case name. On the displayed page, select **Defects** and click **Create and Associate Defects**.
- Associate multiple test cases with bugs: Select the required test cases and click **Associate with Defect**. In the displayed dialog box, select the required bugs and click **OK**.

Step 3 After a defect is created or associated, view the defect information on the **Defects** tab page. You can click  to disassociate the current bug.

----End

Commenting on a Test Case

You can comment on test cases.

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Select a test case to be commented on, click the test case name, and click the **Details** tab.

Step 3 Enter your comments in the **Comments** text box at the bottom of the page and click **Save**.

The comments that are successfully saved are displayed below the **Comments** text box.

----End

Filtering Test Cases

CodeArts TestPlan supports filtering of test cases by using custom filter criteria. The following procedure uses the **Testing Case > Manual Test** as an example.

Using the Default Filter Criteria

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 On the **Manual Test** tab page, select an option from the **All cases** or **All** drop-down list.

- **All cases**: Displays all cases in the current test plan or case library.
- **My cases**: Displays all cases whose **Processor** is the current login user.
- **Unassociated with test suites**: Displays test cases that are not associated with any test suite.

----End

You can click the two drop-down list boxes and filter all cases, your test cases, or test cases that are not associated with test suites.

Setting Advanced Filter Criteria

If the default filter criteria do not meet your requirements, you can customize filter criteria.

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Click **Advanced Filter** above the test case list. Common filter criteria are displayed on the page.

Step 3 Set filter criteria as required and click **Filter**. Test cases that meet the filter criteria are displayed on the page.

You can also click **Save and Filter**. In the displayed dialog box, enter the filter name and click **OK**. The saved filter is added to the **All cases** drop-down list.

Step 4 (Optional) If advanced filter criteria still do not meet requirements, click **Add Filter**, select a filter field from the drop-down list box as required. The filtering field is displayed on the page. Repeat **Step 3** to complete the filtering. You can add custom filter fields for advanced filtering.

----End

Updating Test Case Fields in Batches

Step 1 In the navigation pane, choose **Testing > Testing Case** and click the target case type tab.

Step 2 In the test case list, select the target test cases.

- To select all test cases on the current page, hover the cursor over the topmost check box and click **Select Current Page**.
- To select specified test cases, hover the cursor over the topmost check box and click **Select More**. In the displayed dialog box, select all test cases or set the selection scope, and click **OK**.

Step 3 Click **Update Property**.

Step 4 In the displayed dialog box, select the field to be modified from the left drop-down list box, and select the target option from the right drop-down list box.

- To modify more fields, click **+**.

- To delete a field, click .

----End

Customizing Test Case List Columns

CodeArts TestPlan supports customizing columns to be displayed in the test case list. The following procedure uses manual test cases as an example.

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 On the **Manual Test** tab page, click  in the last column of the test case list. In the displayed dialog box, select the fields to be displayed, deselect the fields to be hidden, and drag the selected fields to rearrange their display sequence. You can also add custom headers to the test case list.

----End

Searching for a Test Case

You can search for test cases by name, ID, or description.

Step 1 Create a test case.

Step 2 In the search box above the test case list, enter a keyword of the name, ID, or description.

Step 3 Click .

Step 4 The required test cases are filtered and displayed in the list.

----End

Deleting a Test Case

In the test case list, you can delete test cases one by one or in batches.

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 In the upper part of the test case list, select the target version or test plan.

Step 3 Click the tab of the target test execution type.

Step 4 Delete test cases one by one or in batches.

- Delete a single test case: Find the target test case, click **...** in the **Operation** column of the test case, and click **Delete**. The test case is removed to the recycle bin.
- Delete test cases in batches: Select the test cases to be deleted and click **Delete** below the list. The test cases are removed to the recycle bin.

If the test case to be deleted is associated with a test suite, a dialog box is displayed. In the dialog box, click  to disassociate the test case from the suite.

----End

Recycle Bin

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Click **Recycle Bin** in the lower left corner of the page.

Step 3 Perform the following operations on the removed test cases as required:

- Click  in the **Operation** column of a test case to restore it.
- Click  in the **Operation** column of a test case to delete it permanently.
- Select some or all test cases and click **Recover** or **Delete** in the lower part of the page to restore or delete them in batches.

----End

7 Creating and Executing a Test Suite

7.1 Creating a Test Suite

In today's fast-iterating product development environment, enterprises face dual pressure to improve software quality and accelerate delivery. Applications on Huawei Cloud often feature diverse functions and complex architectures, resulting in many test cases and greater organization difficulty. Traditional testing methods often lead to scattered test cases, repeated executions, or incomplete coverage, reducing testing efficiency and result reliability.

As a key test management tool, a test suite logically groups related test cases into a set, helping test teams organize and manage tests efficiently. It helps testers systematically organize test activities based on service functions, technical architectures, or test types, ensuring complete test coverage and efficient execution. In addition, it supports flexible combination and reuse, reducing test maintenance costs.

Prerequisites

- Multiple test cases have been created.

Creating a Manual Test Suite

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Execution**. By default, **Test case library** and **Baseline** are displayed.
- Step 3** Click  next to **Test case library** and select a test plan.
- Step 4** On the **Manual Test** tab page, click **Create Suite** in the upper left corner. The creation page is displayed.
- Step 5** Configure the parameters by referring to the following table.

Configuration Item	Mandatory	Description
Name	Yes	Name of a suite. Enter 1 to 128 characters.
Plan Period	No	Select the start time and end time of the test suite.
Description	No	Brief description of the suite. Max. 500 characters.
Tag	No	Enter up to 10 tags separated by spaces.
ID	No	Enter 3 to 128 characters. If this is not set, the service will automatically generate an ID.
Module	No	Select a module in the current project. For details about how to set a module, see Adding Work Item Modules .
Version	No	Enter a version containing at least three characters.
Processor	No	Select a person responsible for processing the suite.

Step 6 Click **Add Case** or **Add Now**, select the test cases to add, and click **OK**.

In the **Add Test Case** dialog box, you can search for the target case by case name, ID, description, or custom field. To filter test cases that have not been added to test suites, click the drop-down list box on the left of the search box and choose **No Associated TestSuite**.

Step 7 Click **Save** to create the manual test suite. The new suite will appear in the suite list on the **Manual Test** tab.

Step 8 (Optional) Click the test suite name to view its details page. Click the  icon in the **Operation** column, select the custom fields to display, and click **OK**.

Step 9 To execute a manual test suite, see [Executing a Manual Test Suite](#).

----End

Creating an Automated API Test Suite

With CodeArts TestPlan, you can create API test suite, run test cases in serial or parallel mode, and accelerate execution of some test cases. The service helps you use resource pools efficiently, reduce task blocking, intercept product defects, and detect issues quickly.

Step 1 Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Execution**.

Step 3 Click  next to **Test case library** and select a test plan.

Step 4 On the **Auto API Test** tab page, click **Create Suite** in the upper left corner. The creation page is displayed.

Step 5 Configure the parameters by referring to the following table.

Configurati on Item	Manda tory	Description
Name	Yes	Name of a suite. Enter 1 to 128 characters.
Preparations	No	Prepare the data required during testing. Click Add Now to select preparation cases from the list.
Followups	No	Release or restore the test data. Click Add Now to select followup cases from the list.
Agent Pool	Yes	The agent pool allows you to access your own execution resources. When executing a task, you can select an agent in the agent pool to execute the task. This improves the task execution efficiency and does not depend on the public resources of CodeArts. Click Agent Pool Management to see existing pools. For details about how to create an agent pool, see Creating an Agent Pool .
ID	No	Enter 3 to 128 characters.
Tag	No	Set tags for the current task as required. The tags are separated by spaces. Each task can be associated with a maximum of 10 tags.
Module	No	Module of the current test suite. The module list comes from the project settings. For details, see Adding Work Item Modules .
Processor	No	Person who needs to complete the test task.
Version	No	Select the version that has been set for the current test plan and test cases.
Description	No	Enter a brief description of the API test suite. Max. 500 characters.
Environmen t Parameters	No	Environment parameters can be referenced by parameters, checkpoints, variables, and URLs of test steps in the entire project. Click  to create an environment parameter.
Max. Duration	Yes	The longest time taken for executing a test case. Cases exceeding this limit will fail. Max. 10 minutes.
Overwrite Test Suite Parameters	No	You can set advanced parameters for a test suite, and overwrite global parameters in a test suite. A maximum of three global parameters can be overwritten.

Configurati on Item	Manda tory	Description
Execute	Yes	There are two types. By default, the task is executed only once. • Once: The test suite is executed only once. • Regularly: After you set an execution frequency, the test suite is executed periodically. Set this parameter to daily.
Start	Yes	There are two options. By default, the task is executed immediately. • Immediately: The task is executed immediately. • At specific time: The task is executed at a specified time.
Sequence	Yes	There are two modes. The default mode is Serial . • Serial: Test cases in the API test suite are executed in serial mode. • Parallel: Test cases in the API test suite are executed in parallel mode.
Case Retry on Failure	No	Number of retries after a test case fails. The value ranges from 0 to 10. The default value is 0.

Step 6 Click **Add Case** or **Add Now** and select the test case to be executed.

In the **Add Test Case** dialog box, you can search for the target case by case name, ID, description, or custom field. To filter test cases that have not been added to test suites, click the drop-down list box on the left of the search box and choose **No Associated TestSuite**.

Step 7 Complete the execution settings as required and click **Save**.

Step 8 Perform the following operations on the created API test suite as required:

- **Modify:** Click the name of the test suite and modify its information.
- **Delete:** Click **...** in the **Operation** column of the test suite and click **Delete**. To delete test suites in batches, select the test suites and click **Delete** at the tool bar below the list.
- **Copy:** Click **...** in the **Operation** column of the test suite and click **Copy Test Suite**. By default, the new suite will be named *Name of the copied suite_copy_Current timestamp*. You can rename the new suite, enter a description, and click **OK** to finish the copy operation.

----End

Quickly Creating a Test Suite in the Test Feature or Requirement Directory

Step 1 In the navigation pane, choose **Testing > Testing Case**.

Step 2 Select the execution mode tab based on the test suite type.

- Step 3** Select the test feature directory or test requirement directory as required.
- Step 4** Click  next to the test feature or requirement.
- Step 5** Click **Create Suite**. The page for creating a test suite of the corresponding type is displayed. For details about how to create test suites of different types, see [Creating a Manual Test Suite](#) and [Creating an Automated API Test Suite](#).
- End

7.2 Executing a Test Suite

In the test execution stage, testers run test suites to check whether the system functions as expected. They record the test results and identify any issues or defects in the product, which assists R&D engineers in analyzing and resolving problems.

Prerequisites

A test suite has been created.

Executing a Manual Test Suite

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Execution**. **Test case library** is displayed by default.
- Step 3** Click  next to **Test case library** and select a test plan.
- Step 4** On the **Manual Test** tab page, locate the test suite to be executed and click  in the **Operation** column.
- Step 5** On the page for executing the manual test suite, set the step result, description, and case result. Switch to other cases in the suite and repeat this step until all cases in the manual test suite are executed.

Automatic roll-up of results in manual test suites: If this function is enabled in [Function Switch](#), a test case in a manual test suite will be automatically set to **Successful** after all test steps of the case are set to **Successful**, and similarly, a test suite will be automatically set to **Successful** after all cases in the suite are **Successful**.

- Step 6** Set the manual test suite result and click **Save** in the upper right corner.
- **Successful**: All test cases in the test suite passed.
 - **Failed**: One or more test steps in the test suite failed.
 - **Pending check**: One or more test steps in the test suite require further verification or confirmation.
 - **Not available**: The test case could not be executed properly.
 - **Blocked**: The test case cannot be executed due to external dependencies or issues.

Step 7 After the execution is complete, click the suite name in the test suite list to view the execution result in the **Execution History** column of the test suite list.

Step 8 Associate existing bugs with the test cases in the suite based on the suite execution result.

1. Open the test suite and click the target case.
2. Click **Associate with Existing Defect**.
3. Select the bugs to be associated.
4. Click **OK**.

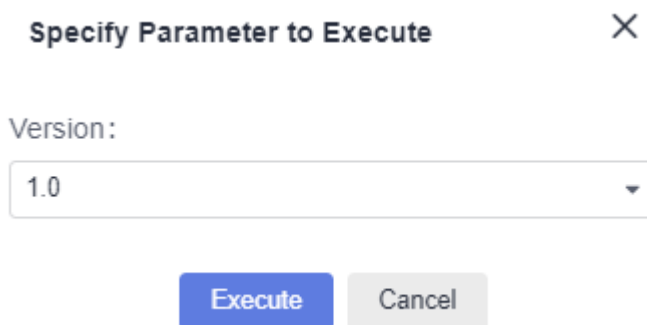
----End

Executing a Manual Test Suite with Parameters

Manual test suites can be executed with parameters.

Step 1 In the test suite list, click ******* in the **Operation** column and select **Specify Parameter to Execute**.

Step 2 In the displayed dialog box, enter the version number and click **Execute**. The execution page is displayed.



The dialog box titled "Specify Parameter to Execute" has a close button (X) in the top right corner. Below the title, there is a label "Version:" followed by a text input field containing "1.0". At the bottom of the dialog, there are two buttons: "Execute" (blue) and "Cancel" (gray).

Step 3 In the drop-down list under the version, set the actual test suite execution result. Enter an actual result, and click **Save**.

Execute : Test Suit01

Version: 1.0

Suite Result

Click to add package execution notes

Click here to choose a filter condition.

Case View Catalog View

1111

Bas

Exe

Step 4 Click the corresponding suite and click the execution history tab to query the execution result.

----End

Executing an Automated API Test Suite

Step 1 Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.

Step 2 In the navigation pane, choose **Testing > Testing Execution**. **Test case library** is displayed by default.

Step 3 Click  next to **Test case library** and select a test plan.

Step 4 Locate the test suite to be executed and click  in the **Operation** column to start automatic execution. After the execution is complete, you can view the execution result in the execution history column.

To stop the suite, click  in the **Operation** column.

Step 5 In the test suite list, click  in the **Operation** column. On the **Execution History** tab page, view the historical execution details of test cases in the test suite.

To create a bug for the test case, click the test case name and click  on the right of the page. To associate the case with a bug, click .

----End

8 Viewing and Evaluating Test Quality

8.1 Viewing the Test Quality Dashboard

CodeArts TestPlan provides dashboards with more than 10 quality metrics, such as the requirement coverage rate, requirement pass rate, case execution rate, and legacy DI, for evaluation of functions, performance, and reliability. You can clearly learn about the progress and risks of the overall test plan through the dashboard.

Project-Level Dashboard

The project dashboard displays statistics on the test case library and test plan. The items include the requirement coverage rate, number of bugs, test case pass rate, test case completion rate, and details of associated bugs.

Quality Report of the Test Case Library

Log in to the CodeArts homepage, search for your target project, and enter the project. In the navigation pane, choose **Testing > Quality Report**. The quality report page is displayed by default.

Report Item	Description
Requirement Coverage	Percentage of tested requirements in the selected sprint and module. This reflects function test coverage. • Not tested: No test case is associated with the requirement, or each associated test case is not complete. • Testing: Some test cases associated with the requirement are uncompleted. • Completed: All test cases associated with the requirement are completed. • Requirement coverage = Completed/Total NOTE In Scrum projects, only statistics on the Feature and Story requirements are collected by default. To collect statistics on the Epic requirements, select epic.

Report Item	Description
Defects	Collects statistics on the total number of defects in the selected sprint and module. The statistics are grouped by defect severity. To monitor a bug, check its group. A closed bug will be grouped into Done. To view or change bug status groups, choose Settings > Work Item > Common Statuses, and locate the Status Category column. NOTE The number of unresolved bugs includes the following: • Number of bugs in the To do and Doing states in a Scrum project. • Number of bugs whose state is not Completed under a Kanban project.
Case Pass Rate	Pass rate of test cases in the selected sprint and module. The test case pass rate and bugs reflect the overall product quality. The statistics are grouped based on the test case result. The test cases that are not executed are included in the Not executed group. Pass rate = Number of successful cases/Total number of cases. Pass rate of executed test cases is calculated based on data from the chosen sprint and module. Pass rate of executed test cases = Number of successful cases/Total number of executed cases.
Key Data Dashboard	Collects statistics on the requirement coverage, total number of defects, case pass rate, and case completion rate associated with manual and API automation test cases in the selected sprint and module. Click  to the right of Manual Test or Auto API Test to view defect or test case details. You can filter items by name or ID, or click a name to open its details page.
Requirement Test	Collects statistics on associated requirements, including total number of associated cases, pass rate, execution rate, and total number of associated bugs. It also displays the corresponding requirement IDs. The requirements can be filtered by status.
Defects Test	Displays a list of associated defects. You can filter the list by defect name or ID. Click a defect name to view its details page. When Test case library is selected, defect statistics are unavailable. To view defect statistics, switch to the corresponding test plan.

Quality Report of a Test Plan

On the **Quality Report** page, click the **Test case Library** drop-down list in the upper left corner and select a test plan. The quality report page of the selected test plan is displayed.

The quality report of a test plan is not filtered by sprint or module. In addition, it shows the test case completion rate and the risk description of the test plan.

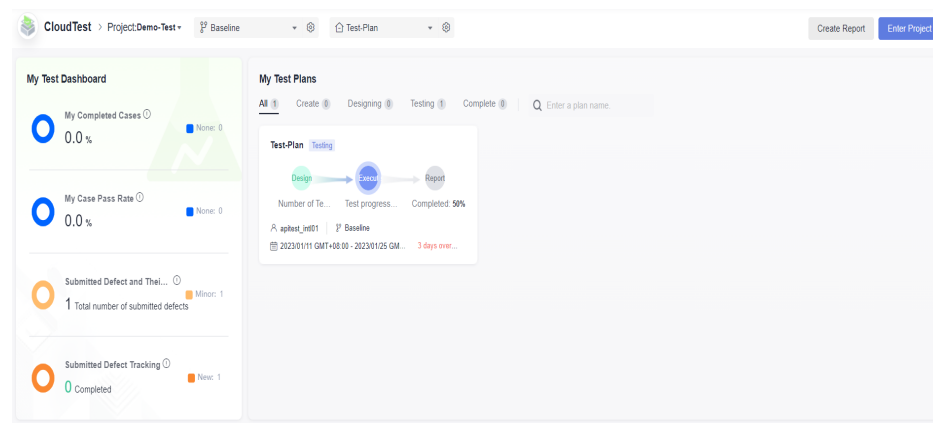
Other report items are the same as those of **the quality report of the test case library**.

Report Item	Description
Case Completion Rate	Number of completed test cases in the selected plan. This reflects the test progress. The statistics are grouped by case status. Completion rate = Number of completed cases/Total number of cases
Risk	Risk of the test plan. You can evaluate the risk level of the test plan and add risk description.
Manual Test	Requirement coverage rate, total number of bugs, pass rate, and completion rate of manual test cases.
Auto API Test	Requirement coverage rate, total number of bugs, pass rate, and completion rate of automated API test cases.

Personal Dashboard

Log in to the CodeArts homepage. On the top navigation bar, choose **Services > TestPlan**. The page displays the personal dashboard. It contains statistics that are processed by the current login user, including:

- Personal case completion rate, case pass rate, submitted bug status, submitted bug severity, bug statistics based on the test case library and test plan in the project
- Personal test plans, test tasks, and submitted bugs based on the test case library and test plan in the project



Custom Test Reports

Create a custom test report in either of the following ways:

- On the homepage
 - a. Log in to the CodeArts homepage. On the top navigation bar, choose **Services > TestPlan**. The homepage is displayed.

- b. Click **Create Report** in the upper right corner, or click **Add Report** in the lower left corner.
 - c. On the displayed **Select Report Type** page, select the type of the report to be created.
- On the quality report page
 - a. Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
 - b. In the navigation pane, choose **Testing > Quality Report**.
 - c. Select a test case library or test plan, select the report type (**Custom Test Report**, **Custom Distribution Report**, and **Progress Test Reports**), and click **Add Report**.
 - d. On the displayed **Select Report Type** page, select the report type as required.

8.2 Creating a Custom Test Report

Custom Test Report

Create a custom test report in either of the following ways:

- On the homepage
 - a. Log in to the service homepage. Click **Create Report** in the upper right corner, or click **Add Report** in the custom test report of the current project in the lower left corner.
 - b. On the displayed **Select Report Type** page, select the report type as required.
- On the quality report page
 - a. Log in to the service homepage, search for your target project, and click the project name to access the project.
 - b. In the navigation pane, choose **Testing > Quality Report**.
 - c. Select the test case library or a test plan, click **Add Report** in the lower left corner of the page, and select the report type as required.

Custom Distribution Reports

Create custom distribution reports with desired filter criteria. The resulting statistics are displayed in the **Quality Report** module.

Step 1 Follow the steps for [creating a custom test report](#), and click **Add Report**.

Step 2 In the displayed dialog box, click **Custom**.

Step 3 Enter a report title.

Step 4 On the right of the page, enter an item name and click **Edit Condition**.

Step 5 In the displayed dialog box, select the resource field name, operator, and resource field value. (To select more fields, click + in the **Operate** column.)

- Step 6** Click **Confirm**. (To add more statistics items, click **Add an Item** and repeat the previous step.)
- Step 7** In the **Test Features** directory, select the required directory.
- Step 8** Click **Refresh Report**.
- Step 9** (Optional) To reuse this report configurations later, click **Save as Template** in the upper right corner of the page, enter the template name, and click **OK**.
- Step 10** Click **Save**.
- End

Creating a Progress Report

Create a progress report to view the number and execution status of test cases for a specific time period.

- Step 1** Follow the steps for [creating a custom test report](#), and click **Add Report**.
- Step 2** In the displayed dialog box, click **Progress Report**.
- Step 3** Enter a report title, and select a report period and a test feature directory.
- Step 4** Click **Save**. The progress report displays the number and execution status of test cases within the selected date range.
- End

8.3 Evaluating Test Quality

Custom test reports let you flexibly evaluate test quality across requirement and defect completion, test case execution, and risk assessment. You can download reports in multiple formats to efficiently share test progress.

Notes and Constraints

Up to 15 attachments, max. 10 MB each, can be uploaded to **Custom Information** of a test report.

Creating a Test Report

On the Quality Report Page

- Step 1** In the navigation pane, choose **Testing > Quality Report**.
- Step 2** In the upper part of the page, select the target version and test plan from the drop-down lists. (**Test case library** and **Baseline** are selected by default.)
- Step 3** Click **Generate Report**.
- Step 4** In the displayed dialog box, enter the report name (default format: *Project name_Selected version name_Selected test plan name_Random code generated by the system*), select a version (defaults to the version selected on the quality report page), test plan (defaults to the plan selected on the quality report page. You can select multiple test plans), test conclusion, and reviewer.

- Step 5** Click **OK**.
- Step 6** After the creation is complete, choose **Testing > Quality Assessment** to view the created test report.

----End

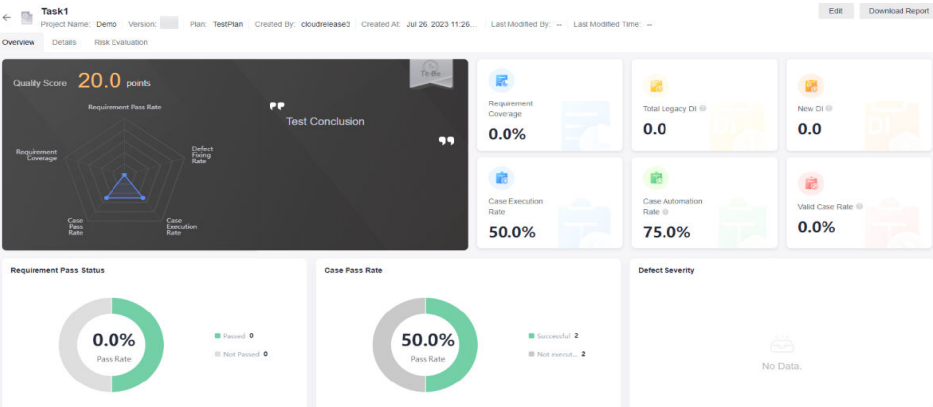
On the Quality Assessment Page

- Step 1** Log in to the CodeArts homepage, search for your target project, and enter the project.
- Step 2** In the navigation pane, choose **Testing > Quality Assessment**.
- Step 3** Click **Generate Report** in the upper left corner of the page. Enter a report name, select a version, test plan (or multiple test plans), and reviewer, enter a test conclusion and description, and click **OK**.

----End

Overview

Click the report name. The **Overview** page of the test report is displayed by default. The **Overview** page displays the test data of the selected test plan.



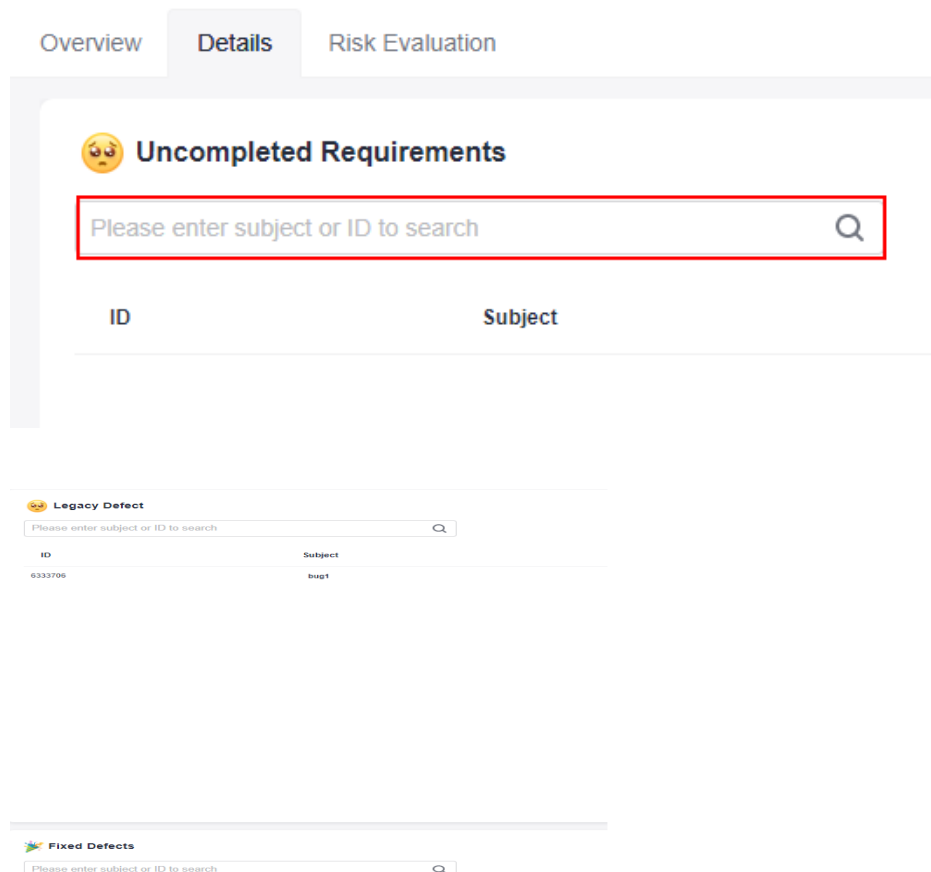
Report Item	Description
Requirement Coverage (of all requirements)	Percentage of tested requirements in selected plans. This reflects the test coverage of function points. Completed requirements: Requirements where all associated test cases are marked as completed. Total requirements: The total number of requirements associated with the test plan. Requirement coverage = Number of completed requirements/ Total number of requirements

Report Item	Description
Project Total Legacy DI	Weighted sum of uncompleted defects in the project, calculated by severity. The recommended formula is as follows: $DI = \text{Critical} \times 10 + \text{Major} \times 3 + \text{Minor} \times 1 + \text{Trivial} \times 0.1$ NOTE All defects are calculated based on their severity levels. The DI value of each defect is as follows: Trivial: 0.1 Minor: 1 Major: 3 Critical: 10
New DI	Weighted sum of uncompleted defects that are newly added to the test plan. This includes defects directly associated with the test plan and those associated with its test cases. Defects are weighted by severity using the following recommended formula: $DI = \text{Critical} \times 10 + \text{Major} \times 3 + \text{Minor} \times 1 + \text{Trivial} \times 0.1$
Case Execution Rate	Statistics on the execution of test cases. Case execution rate = Number of test cases that are executed as planned and have execution results/Total number of test cases as planned
Case Automation Rate	Proportion of automated test cases. Case automation rate in the test plan = (Total number of test cases – Number of manual test cases)/Total number of test cases
Valid Case Rate	Proportion of executed test cases where defects are found to all executed test cases. Valid case rate = Number of executed test cases where defects are found/Number of executed test cases
Requirement Pass Status	Passed requirements: Requirements whose associated test cases have all passed and been completed Requirements not passed: Requirements whose associated test cases have not all passed and been completed Requirement pass rate = Number of passed requirements / (Number of passed requirements + Number of requirements not passed)
Case Pass Rate	The pass rate reflects product quality. The pass rate covers all test cases in selected plans. Another chart displays data by result. Pass rate = Number of successful cases/Total number of cases
Defect Severity	Displays the number of defects associated with the selected plan.
Automation Rate	Displays the proportion of automated test cases in a pie chart. Case automation rate in the test plan = (Total number of test cases – Number of manual test cases)/Total number of test cases
Case Pass Rate by Type	Displays the pass rate of test cases of different test types.
Bugs by Module	Displays the number of defects by module.

Details

On the **Details** page, you can view the outstanding and completion status of requirements and defects in the test plan.

Enter a title or number in the search box to search for the corresponding requirement or defect.



Evaluating Risks

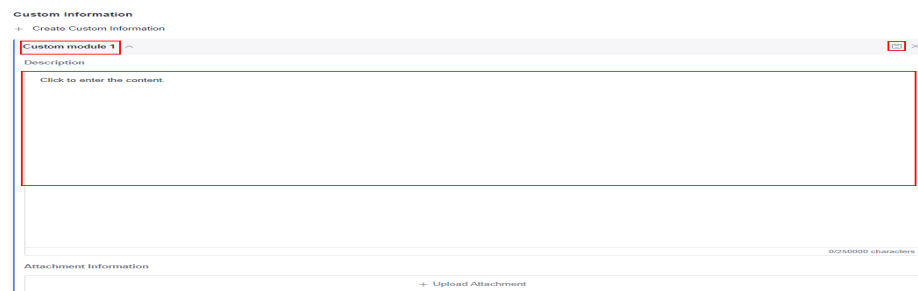
On the **Risk Evaluation** page, enter your evaluation of the test quality and risks, or add custom information modules to the report.

- **Quality attribute evaluation**
Select a test type on the top, click the text box area, enter evaluation information, and click **Save**.
- **Risk analysis**
Evaluate risks based on the test progress. Click the text box, enter risk analysis information, and click **Save**.
- **Custom information**
Click **Create Custom Information** and add an information module.

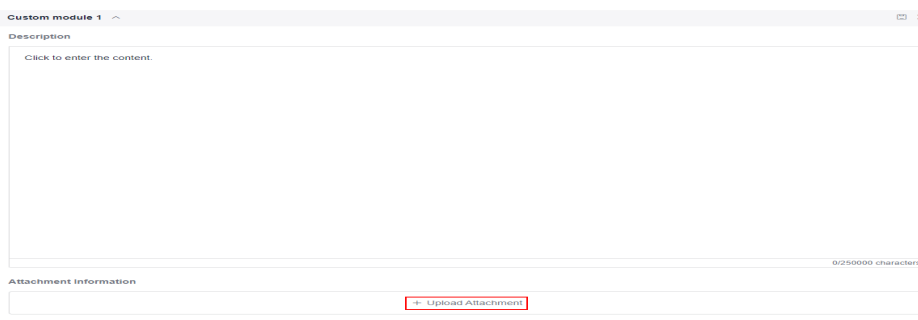
Custom Information

+ Create Custom Information

Click the module name to redefine the module title, enter the related description, and click  to save the modification. To delete the module, click .



Click **Upload Attachment** to upload a local file to the customized module.

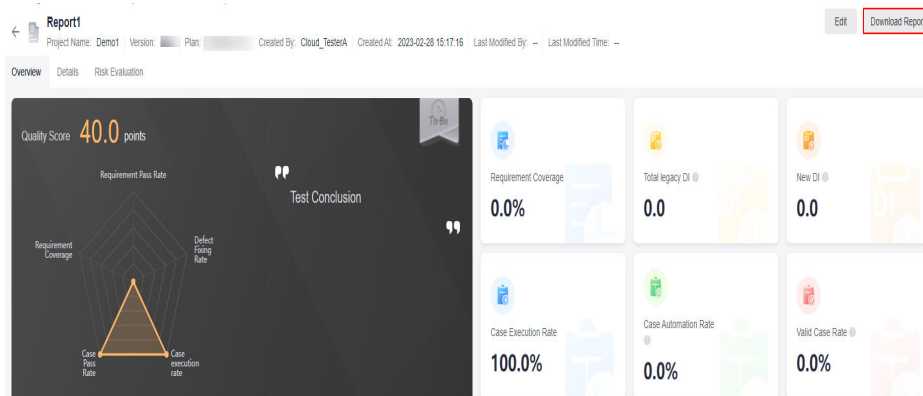


NOTE

1. The size of a single file to be uploaded cannot exceed 10 MB.
2. A maximum of 15 attachments can be uploaded.

Downloading a Report

In the upper right corner of the page, click **Download Report** to download the assessment report in PDF or Word format to the local PC.



9 Managing CodeArts TestPlan Project Notifications

9.1 Setting System Messages

Scenarios

CodeArts TestPlan sends a system message to specified recipients when any of the following events occurs.

TestPlan

Table 9-1 Event types

Event Type		Recipient
Test case	Assign Case Handler	Processor
	Execute Manual Test Case	• Test case creator • Test case handler
	API Test Case Execution Completed	Operator
	Update Case	• Test case creator • Test case handler • Project manager • Test manager • Project creator
	Delete Case	
Test suite	Create Test Suite	Test suite handler
	Update Test Suite	• Test suite creator • Test suite handler

Event Type		Recipient
	Delete Test Suite	• Test suite creator • Test suite handler • Project manager • Test manager • Project creator
	Test Suite Execution Completed	Test suite executor
Test report	Create Report	• Project manager • Test manager • Project creator
	Update Report	• Report creator
	Delete Report	• Project manager • Test manager • Project creator
Branch plan	Create Plan	• Plan creator
	Delete Branch	• Project manager • Test manager • Project creator
	Change Plan	• Plan creator
	Edit Risk Info (on the test quality dashboard)	• Plan handler • Project manager
	Delete Plan	• Test manager • Project creator
	Send Quality Report upon Test Plan Completion	• Project creator
Directory	• Delete Test Feature Directory	• Directory creator • Project manager • Test manager • Project creator
Comment notification	• Comment Notification (comment changes in manual test cases)	@Mentioned member

Prerequisites

You have created a CodeArts project, and your role in the project is **Project Administrator** or **Project Manager**.

Setting System Messages

- Step 1** Go to the CodeArts homepage.
- Step 2** Click the target project name to go to the project.
- Step 3** In the navigation pane, choose **Settings > General > Notifications**.
- Step 4** Click the **System Message** tab.
- Step 5** Specify recipients for each event type and determine whether to enable notifications.

The updated configuration is displayed.

----End

9.2 Setting Email Notifications

Scenarios

CodeArts TestPlan sends an email to the specified recipients when any of the following events occurs.

The recipients will receive emails only if they have enabled **Email Notifications** and set a valid email address on the **This Account Settings > Notifications** page.

TestPlan

Table 9-2 Event types

Event Type		Recipient
Test case	Assign Case Handler	Test case handler
	Update Case	• Test case creator • Test case handler • Project manager • Test manager • Project creator
	Execute Manual Test Case	• Test case creator • Test case handler • Test plan handler
	API Test Case Execution Completed	Test case executor

Event Type		Recipient
	Delete Case	• Test case creator • Test case handler • Test plan handler • Project manager • Test manager • Project creator
Test suite	Create Test Suite	Test suite handler
	Update Test Suite	• Test suite creator • Test suite handler
	Delete Test Suite	• Test suite creator • Test suite handler • Project manager • Test manager • Project creator
	Test Suite Execution Completed	• Test suite executor
Test report	Create Report	• Project manager • Test manager • Project creator
	Update Report	• Report creator
	Delete Report	• Project manager • Test manager • Project creator
Branch plan	Create Plan	• Plan handler • Project manager • Test manager • Project creator
	Change Plan	• Plan creator
	Edit Risk Info (on the test quality dashboard)	• Plan handler • Project manager
	Delete Plan	• Test manager
	Send Quality Report upon Test Plan Completion	• Project creator

Event Type		Recipient
	Delete Branch	• Branch creator • Project manager • Test manager • Project creator
Directory	Delete Test Feature Directory	• Directory creator • Directory handler • Project manager • Test manager • Project creator
Comment notification	Comment Notification (comment changes in manual test cases)	@Mentioned member

Prerequisites

You have created a CodeArts project, and your role in the project is **Project Administrator** or **Project Manager**.

Setting Email Notifications

- Step 1** Go to the CodeArts homepage.
- Step 2** Click the target project name to go to the project.
- Step 3** In the navigation pane, choose **Settings > General > Notifications**.
- Step 4** Click the **Email** tab.
- Step 5** Specify recipients for each event type and determine whether to enable notifications.

The updated configuration is displayed.

----End

9.3 Setting WeCom Subscriptions

Scenarios

When any of the following events occurs, CodeArts TestPlan sends a message through HTTP to the WeCom group that has subscribed to the event.

TestPlan

Table 9-3 Event types

Category	Event Type
Test case	Assign Case Handler
	Update Case
	Execute Manual Test Case
	API Test Case Execution Completed
	Delete Case
Test suite	Create Test Suite
	Update Test Suite
	Delete Test Suite
	Test Suite Execution Completed
Test report	Create Report
	Update Report
	Delete Report
Branch plan	Create Plan
	Change Plan
	Edit Risk Info (on the test quality dashboard)
	Delete Plan
	Delete Branch
	Send Quality Report upon Test Plan Completion
Directory	Delete Directory
Comment notification	Comment Notification (comment changes in manual test cases)

Prerequisites

You have created a CodeArts project, and your role in the project is **Project Administrator** or **Project Manager**.

Setting WeCom Subscriptions

Step 1 Go to the CodeArts homepage.

- Step 2** Click the target project name to go to the project.
- Step 3** In the navigation pane, choose **Settings > General > Notifications**.
- Step 4** Click the **WeCom** tab.
- Step 5** Configure the required parameters and click **OK**.

Table 9-4 Setting WeCom Subscriptions

Parameter	Description
URL	The webhook URL of your WeCom group robot. It must start with http:// or https:// . The URL cannot exceed 256 characters. For details about how to obtain the webhook URL of your WeCom group robot, see section "How to set group robot" in the WeCom Help Center. (The section name is subject to updates of the WeCom Help Center.)
Event Type	Event type that can trigger a WeCom notification. For details about the supported event types, see Scenarios .
Content	Content of the WeCom notification. For details about the configurable content, see Scenarios . This parameter is optional. Configure it as needed.
@user_id	WeCom user IDs of the members you want to mention. Separate the IDs with semicolons (;). The total length of all IDs cannot exceed 1,000 characters. Digits, letters, semicolons (;), periods (.), hyphens (-), underscores (_), and at signs (@) are supported. This parameter is optional. Configure it as needed.

The updated configuration is displayed.

----End

9.4 Setting DingTalk Subscriptions

Scenarios

When any of the following events occurs, CodeArts TestPlan sends a message through HTTP to the DingTalk group that has subscribed to the event.

TestPlan

Table 9-5 Event types

Category	Event Type
Test case	Assign Case Handler
	Update Case
	Execute Manual Test Case
	API Test Case Execution Completed
	Delete Case
Test suite	Create Test Suite
	Update Test Suite
	Delete Test Suite
	Test Suite Execution Completed
Test report	Create Report
	Update Report
	Delete Report
Branch plan	Create Plan
	Change Plan
	Edit Risk Info (on the test quality dashboard)
	Delete Plan
	Delete Branch
	Send Quality Report upon Test Plan Completion
Directory	Delete Directory
Comment notification	Comment Notification (comment changes in manual test cases)

Prerequisites

You have created a CodeArts project, and your role in the project is **Project Administrator** or **Project Manager**.

Setting DingTalk Subscriptions

Step 1 Go to the CodeArts homepage.

- Step 2** Click the target project name to go to the project.
- Step 3** In the navigation pane, choose **Settings > General > Notifications**.
- Step 4** Click the **DingTalk** tab.
- Step 5** Configure the required parameters and click **OK**.

Table 9-6 Configuring DingTalk notifications

Parameter	Description
URL	The webhook URL of your DingTalk robot. It must start with http:// or https:// . The URL cannot exceed 256 characters.
Event Type	Event types that can trigger a DingTalk notification. For details about the supported event types, see Scenarios .
Content	Content of the DingTalk notification. For details about the configurable content, see Scenarios . This parameter is optional. Configure it as needed.
@user_id	DingTalk user IDs of the members you want to mention. Separate the IDs with semicolons (;). The total length of all IDs cannot exceed 1,000 characters. Digits, letters, semicolons (;), periods (.), hyphens (-), underscores (_), and at signs (@) are supported. This parameter is optional. Configure it as needed.
@Mobile numbers	Mobile numbers of the members you want to mention. Separate mobile numbers with semicolons (;). The total mobile numbers cannot exceed 1,000 characters. Digits and semicolons (;) are supported.
Additional Signature	If your DingTalk robot has enabled signature-based encryption in security settings, enter the signing secret with a maximum of 2,000 characters.

The updated configuration is displayed.

----End

9.5 Setting Feishu Subscriptions

Scenarios

When any of the following events occurs, CodeArts TestPlan sends a message through HTTP to the Feishu group that has subscribed to the event.

TestPlan

Table 9-7 Event types

Category	Event Type
Test case	Assign Case Handler
	Update Case
	Execute Manual Test Case
	API Test Case Execution Completed
	Delete Case
Test suite	Create Test Suite
	Update Test Suite
	Delete Test Suite
	Test Suite Execution Completed
Test report	Create Report
	Update Report
	Delete Report
Branch plan	Create Plan
	Change Plan
	Edit Risk Info (on the test quality dashboard)
	Delete Plan
	Delete Branch
	Send Quality Report upon Test Plan Completion
Directory	Delete Directory
Comment notification	Comment Notification (comment changes in manual test cases)

Prerequisites

You have created a CodeArts project, and your role in the project is **Project Administrator** or **Project Manager**.

Setting Feishu Subscriptions

Step 1 Go to the CodeArts homepage.

- Step 2** Click the target project name to go to the project.
- Step 3** In the navigation pane, choose **Settings > General > Notifications**.
- Step 4** Click the **Feishu** tab.
- Step 5** Configure the required parameters and click **OK**.

Table 9-8 Setting Feishu subscriptions

Parameter	Description
URL	The webhook URL of your Feishu robot. It must start with http:// or https:// . The URL cannot exceed 256 characters. For details about how to obtain the webhook URL of your Feishu robot, see section "Custom bot usage guide" in the Feishu Open Platform Help Center. (The section name is subject to updates of the Feishu Help Center.)
Event Type	Event type that can trigger a Feishu notification. For details about the supported event types, see Scenarios .
Content	Content of the Feishu notification. For details about the configurable content, see Scenarios . This parameter is optional. Configure it as needed.
@user_id	Feishu user IDs of the members you want to mention. Separate the IDs with semicolons (;). The total length of all IDs cannot exceed 1,000 characters. Digits, letters, semicolons (;), periods (.), hyphens (-), underscores (_), and at signs (@) are supported. This parameter is optional. Configure it as needed.
Additional Signature	If your Feishu robot has enabled signature-based encryption in security settings, enter the signing secret with a maximum of 2,000 characters. For details about how to obtain the signing secret of your Feishu robot, see section "Custom bot usage guide" in the Feishu Open Platform Help Center. (The section name is subject to updates of the Feishu Help Center.)

The updated configuration is displayed.

----End

10 Settings

10.1 Function Switch

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Settings > Testing**.
- Step 3** Click the **Function Switch** tab and enable or disable the following functions as required.
- **Requirement update dot:** Updated requirements are indicated by a red dot next to their names in the test requirement directory on the test case page. Click the red dot to see the change history. For details, see the instructions for [associating test cases with requirements](#).
 - **When importing a swagger file, use the fields defined by the API as the script template name:** The default script template name is the value of **operationId**. You can turn on the switch to use the value of **summary**.
 - **Interface automation testing supports redirection.:** redirection during execution of automated API test cases.
 - **Test design generates test cases step by step:** Generate test cases on a mind map, with each step having an expected result. For details, see the instructions for [step-by-step case generation](#).
 - **Automatic roll-up of manual test suite results:** If all the steps in a test case are **Successful**, the test case result will be automatically set to **Successful**; similarly, if all the test cases within a test case suite are **Successful**, the suite result will be automatically set to **Successful**.

----End

10.2 Managing Project Members

Project members who have logged in to and accessed the CodeArts TestPlan pages are displayed in the CodeArts TestPlan user list. The project creator can manage the user list as required.

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Settings > Testing**.
- Step 3** Click the **User Management** tab. On the displayed tab page, view users who have used CodeArts TestPlan. The project creator can delete users as required.
- End

10.3 Setting Test Case Fields

If the preset test case fields do not meet your actual requirements, you can add custom fields.

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Settings > Testing**.
- Step 3** Click the **Test Case Settings** tab. Test case fields are displayed.
- Step 4** Configure the fields:
- Click  in the **Operation** column to modify the names of some fields, add or delete options, and enable or disable **Show** or **Required**.
 - Enable or disable **Display** or **Required** on the setting page.
 - Click the drop-down list box of the default value of a field and select a value as the default value of the field after a test case is created.
 - Click  in the **Operation** column to delete a field.
- End

Adding a Custom Field

You can set custom fields. Custom fields are displayed in the details of test cases.

- Step 1** In the **Test Case Settings** tab page, click **Add Field**. Max. 25 custom fields can be added.
- Step 2** Set **Field Name**, **Description**, and **Field Type**, enable or disable **Show** and **Required**, and click **OK**.
- **Single-line text**: Only one line can be entered in this field. The maximum length is 100 characters.
 - **Multi-line text**: Multiple lines of fields can be entered in this field. The maximum length is 2,000 characters.
 - **Single-choice list**: Only one option can be selected in this field. You can set up to 300 options, each within 40 characters. Options can be added by either of the following ways:
 - Click **Add Option** and enter the new option.
 - Click **Multiple Entries**. In the displayed dialog box, enter an option and press **Enter**. Each line of text will generate one option. Click **OK**.

- **Multi-choice list:** Multiple options can be selected in this field. You can set up to 100 options, each within 40 characters. Options can be added by either of the following ways:
 - Click **Add Option** and enter the new option.
 - Click **Multiple Entries**. In the displayed dialog box, enter an option and press **Enter**. Each line of text will generate one option. Click **OK**.
- **Date Time:** Field for setting date and time.
- **Date:** Field for setting date.
- **Number:** Field for entering an integer ranging from -9999999999 to 9999999999.
- **Decimal:** Field for entering a decimal up to two decimal places.
- **Single-choice user:** Field for selecting a user from the drop-down list box.
- **Multi-choice user:** Field for selecting multiple users from the drop-down list box.

Step 3 The created field is displayed in the list. To edit the field, click . To delete the field, click .

----End

Tenant Template Fields

Tenant administrators can configure test case fields for the entire tenant, enabling all users to reuse them across projects and reduce repetitive configuration.

NOTE

Only tenant administrators (IAM users with the **Tenant Administrator** permissions) can create and update tenant template fields.

Step 1 Go to the **Test Case Settings** page. In the upper right corner of the page, hover over the  icon next to **Tenant Template Fields**.

Step 2 Click **to set**.

Step 3 Configure fields by referring to [Step 4](#) and [Adding a Custom Field](#), and then return to the test case settings page.

Step 4 Click the project list in the upper left corner of the page and switch to another project.

Step 5 On the test case settings page, click **Tenant Template Fields**. The configured fields will overwrite the system fields of the project.

If you repeat the preceding steps to update the tenant template fields, delete all custom fields before applying the updated tenant template and click **Tenant Template Fields**.

----End

10.4 Test Suite Status and Results

If the preset test suite status and result do not meet your actual requirements, you can add other status and results as required.

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Settings > Testing**.
- Step 3** On the **Test Suite Settings** tab page, click **Add Status** or **Add Result**, and configure the new suite status and result as required.

----End

10.5 Background Images and Logos of Test Reports

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Settings > Testing**.
- Step 3** Replace the images of cover page background, body page background, and logo of the test report with the images from your local host.
- Cover page background image: The recommended format is JPG or PNG; the recommended size is 794 x 1123 pixels.
 - Body page background image: The recommended format is JPG or PNG; the recommended size is 794 x 1123 pixels.
 - Logo: The recommended format is JPG or PNG; the recommended width is ≤ 90 pixels and the recommended height is ≤ 40 pixels.
- Step 4** After the images are uploaded, click **Save**.
- Step 5** [Download a report](#) and view the reconfigured style.

----End

10.6 Request Timeout Interval, Resource Pool, and DNS Mapping

After DNS mapping is configured, if the URL request path in the automated API test case is a domain name, the IP address corresponding to the request is automatically mapped.

You can set the request timeout interval, agent pool, and DNS mapping in **Settings**.

- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Settings > Testing > Other**.

Step 3 Select the target resource pool for API case debugging and click **OK**.

- **Default Resource Pool:** **DEFAULT** is the default executor provided by CodeArts TestPlan.
- **Custom Resource Pool:** Add your own executors to CodeArts for unified, centralized scheduling of automated test tasks. You must first [create a resource pool](#).

(Optional) **The executor uses host network resources:** If this function is enabled, the executor will use the network configuration of the host where the executor is located to obtain external resources.

Step 4 (Optional) **User-defined executor parameter path:** Enter the full path of the parameter configuration file that can be obtained by the executor. Use the full path format, for example, **/etc/sample_enc**. Ensure that the file exists. Otherwise, the custom executor task will fail.

 NOTE

1. The parameter configuration file is stored on the host where the executor is located. If there are multiple nodes, this file must be available on each host.
2. Currently, this configuration item supports only proxy configuration in the *Key=value* format. Examples (**http_proxy**, **https_proxy**, and **no_proxy** are in lowercase):
 - http_proxy=http://proxy.example.com:8080
 - https_proxy=https://proxy.example.com:8080
 - no_proxy=localhost,127.0.0.1
3. If the proxy contains special characters (such as ^ and %), convert them by URL encoding.
 - For example, if the proxy address is **http://TestU^&*ser:test%^&@Passproxy.example.com:8080**, in which **TestU^&*ser** is the account, and **test%^&** is the password. After being URL-encoded, the account is converted to **TestU%5E&*ser**, and the password to **test%25%5E&**. Then the proxy address to be configured is **http://TestU%5E&*se:test%25%5E&@Passproxy.example.com:8080**.

Step 5 Set **Request Timeout Interval**. The default value is 10,000 ms.

Step 6 Set **DNS Mapping**.

- **Domain Name:** a unique website identifier
- **IPS:** the location of a device or host, containing four segments of numbers ranging from 0 to 255, separated by periods (.)

Step 7 Enter a JSON array to mask your sensitive parameters. For example, **["password","psd"]**. This example configuration will mask the middle six digits of a password if it has more than six digits, or mask all the digits if it has six or fewer digits.

----End

11 Querying Audit Logs

Cloud Trace Service (CTS) records operations on CodeArts TestPlan for query, audit, and backtrack.

Operations Recorded by CTS

Table 11-1 CodeArts TestPlan operations recorded by CTS

Operation	Resource Type	Trace Name
Merge cases to the baseline	Testcase	mergeTestCase
Import test cases from other branches	Testcase	importTestCase
Download the import template	Testcase	downloadImportTemplate
Import test cases from a file	Testcase	importTestCaseListTask
Export test cases	Testcase	exportTestCases
Export incorrect test cases	Testcase	exportErrorTestCases
Download the exported case file	Testcase	downloadExportTestCasesFile
Synchronize test cases	Testcase	syncTestCases
Add a test case comment	Testcase	addTestCaseComment
Update a test case comment	Testcase	updateTestCaseComment
Delete a test case comment	Testcase	deleteTestCaseComment

Operation	Resource Type	Trace Name
Create test case reviews in batches	Testcase	createTestCaseReview-Batch
Update a test case review	Testcase	updateTestCaseReview
Delete test case reviews in batches	Testcase	deleteTestCaseReview-Batch
Download an attachment	Testcase	downloadAttachment-FromObs
Upload attachment cache	Testcase	uploadCacheFile
Delete attachment cache	Testcase	deleteCacheFile
Upload an image	Testcase	uploadStepImg
Download an image	Testcase	downloadStepImage
Download an attachment	Testcase	downloadAttachment
Upload a local attachment	Testcase	uploadLocalAttachment
Add an attachment	Testcase	addAttachments
Delete an attachment	Testcase	deleteAttachment
Query test report template information	TestReport	getBackgroundInfo
Update the test report template	TestReport	uploadBackground
Download the background image of the test report template	TestReport	downloadBackground
Update the custom display configurations of test cases	Testcase	updateCustomizedConfig
Add a custom filter	Testcase	addCustomizedFilter
Update a custom filter	Testcase	updateCustomizedFilter
Delete a custom filter	Testcase	deleteCustomizedFilter
Create a test report	TestReport	createReport
Update a test report	TestReport	updateReport
Delete a test report	TestReport	deleteReport

Operation	Resource Type	Trace Name
A tenant joins a trial	Domain	addDomainTrial
Stop a debugging task	APITestSchedule	stopCaseTask
Create a test task	APITestSchedule	createTestCasesTask
Create a case	testcase	createTestCaseAndScript
Create a test suite task	APITestSchedule	createTestSuiteTask
Update a case	testcase	updateTestCaseAndScript
Obtain a sensitive variable	variable	getSensitiveVariable
Refresh a script	testcase	refreshJavaCode
Create cases by importing a file	testcase	createTestcasesByFile
Add a custom URL keyword	basic-aw	addUserDefinedUrlKey-Word
Create a keyword directory	basic-aw-cata	createAwCata
Delete keywords in batches	basic-aw	batchDeleteBasicAW
Delete a case	delete-test-case	deleteTestCase
Import cases from Postman	testcase	importTestCaseFromPostman
Process variables in batches	variable	batchDealVariable
Delete case associations	testcase	deleteTestCaseRelation
Delete the arts keywords in batches	basic-aw	batchDeleteBasicAW-arts
Obtain a sensitive variable	testcaseVariable	getSensitivePropertybyId
Update a keyword	basic-aw	updateAPI-arts
Create a keyword	testcase	createKeyword
Update notification configurations	NoticeConfigs	updateNoticeConfigs
Create a combined keyword	basic-aw	createCombinedAw
Import test cases	testcase	importTestCase

Operation	Resource Type	Trace Name
Obtain sensitive information	testcaseVariable	getSensitiveProByTmssl-dAndName
Create a project	project	createProject
Delete a test case and its script	testcase	deleteTestCaseAndScript
Delete a keyword directory	basic-aw-cata	deleteAwCata
Delete case cache	testcaseCache	deleteTestcaseCache
Keyword sorting	basic-aw	orderBasicAW
Create a TMSS case	testcase	createTMSSCaseAndCopyScript
Upload a file	upload-file	uploadFile
Update a keywords directory	basic-aw-cata	updateAwCata
Copy a tenant keyword	basic-aw	copyCombinedAw
Create a case by Swagger	testcase	createTestcasesBySwagger
Add an APIArts keyword	basic-aw	addAPI-arts
Obtain sensitive information	basic-aw	getSensitiveValueByAwIdAndName
Delete a keyword	basic-aw	deleteBasicAW
Create a test suite	testsuit	createTestSuit
Download a template	download-template	downloadTemplate
Send a notification	noticeConfig	sendNotice
Update a keyword name	AwNameView	updateAwNameView
Modify a keyword directory	basic-aw-cata	modifyAwCata
Update the timeout interval	TimeOutSet	updateTimeOutSet
Add a task	Task	addTask
Modify a task	Task	updateTask
Delete a task	Task	deleteTask
Start a task	Task	startTask
Stop a task	Task	stopTask

Operation	Resource Type	Trace Name
Update intelligent alarms	Task	updateTaskWiseAlert
Enable or disable alarms	Task	pauseOrResumeAlert
Favorite a task	Task	favoriteTask
Unfavorite a task	Task	cancelFavoriteTask
Add an alarm group	AlertGroup	addGroup
Update an alarm group	AlertGroup	updateGroup
Update the alarm group data	AlertGroup	updateProperties
Delete an alarm group	AlertGroup	delete
Add an alarm template	AlertTemplate	addTemplate
Update an alarm template	AlertTemplate	updateTemplate
Delete an alarm template	AlertTemplate	delete
Ignore an alarm	Msg	ignore
Create a mind map	project	saveMindmap
Import a mind map	mindmap	importMindmap
Edit a mind map	mindmap	editMindmap
Delete a mind map	mindmap	deleteMindmapByld
Upload a file	mindmap	saveFileLibFile
Create a mind map backup	mindmap	mindmapBackupBackUp-Mindmap
Delete a mind map backup	backup	deleteMindmapBackup-Byld
Restore a mind map	mindmap	mindmapBackupRecover-Mindmap
Delete a mind map from the recycle bin	recycle	deleteMindmapRecycle-Byld
Restore a mind map from the recycle bin	mindmap	mindmapRecycleRecoverMindmap
Save a review	mindmap	saveReview
Delete reviews in batches	review	logicDeleteReviewByNo-deld

Operation	Resource Type	Trace Name
Delete a review	node	logicDeleteReviewById
Update a review	review	updateReview
Save a template	mindmap	saveTemplate
Delete a template	template	deleteTemplateById
Update a template	template	updateTemplate
Delete a feature directory	feature	deleteFeature

Viewing Audit Logs

Query CodeArts TestPlan traces on the CTS console. For details, see [Querying Real-Time Traces](#).