

**CodeArts TestPlan**

# **TestPlan User Guide**

**Issue**            01  
**Date**             2023-11-30



**Copyright © Huawei Technologies Co., Ltd. 2023. 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 Technologies Co., Ltd.

## **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are trademarks 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 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.

# Security Declaration

## Vulnerability

Huawei's regulations on product vulnerability management are subject to "Vul. Response Process". For details about the policy, see the following website:<https://www.huawei.com/en/psirt/vul-response-process>  
For enterprise customers who need to obtain vulnerability information, visit:<https://securitybulletin.huawei.com/enterprise/en/security-advisory>

---

# Contents

---

<b>1 Service Entries.....</b>	<b>1</b>
<b>2 Testing Plan.....</b>	<b>2</b>
2.1 Introduction.....	2
2.2 Creating and Editing a Test Plan.....	2
2.3 Designing and Executing a Test Plan.....	4
2.4 Managing and Measuring a Test Plan.....	7
<b>3 Testing Design.....</b>	<b>9</b>
3.1 Introduction.....	9
3.2 Mind Maps.....	9
3.2.1 Creating a Mind Map.....	9
3.2.2 Drawing a Mind Map.....	11
3.2.3 Managing a Mind Map.....	13
3.2.4 Managing Templates.....	18
3.3 Requirement Catalog Test Design Strategies.....	19
3.3.1 Creating a Mind Map Based on Requirements.....	20
3.3.2 Designing a Scenario.....	20
3.3.3 Designing a Test Point.....	21
3.3.4 Designing a Test Case.....	22
3.3.5 Generating a Test Case.....	23
3.3.6 Adding a Tag and Remarks.....	26
3.3.7 Performing Online Review.....	27
3.3.8 Collecting Node Statistics.....	28
3.4 Feature Catalog Test Design Strategies.....	29
3.4.1 Creating a Mind Map Based on Features.....	29
3.4.2 Associated with a Requirement.....	29
<b>4 Testing Management.....</b>	<b>31</b>
4.1 Introduction.....	31
4.2 Creating, Editing, and Deleting a Version.....	32
4.3 Managing a Version.....	33
4.4 Measuring a Version.....	35
<b>5 Testing Case.....</b>	<b>37</b>
5.1 Introduction.....	37

5.2 Test Case Library.....	38
5.3 Manual Test Cases.....	38
5.3.1 Creating a Manual Test Case.....	39
5.3.2 Creating a Test Procedure.....	41
5.3.3 Migrating a Manual Test Case.....	41
5.3.4 Executing a Manual Test Case.....	43
5.4 API Automation Test Cases.....	45
5.4.1 Creating an API Automation Test Case.....	45
5.4.2 Writing an API Automation Script.....	46
5.4.3 Keyword Library.....	48
5.4.4 API Keywords.....	50
5.4.5 Combined Keywords.....	54
5.4.6 Importing cURL to Generate Test Scripts.....	57
5.4.7 Setting an API Request.....	58
5.4.8 Setting a Test Checkpoint.....	62
5.4.9 Setting Response Extraction.....	75
5.4.10 Importing a Postman File.....	77
5.4.11 Inserting Logic Control.....	79
5.4.12 Setting Test Case Parameters.....	81
5.4.13 Setting Environment Parameters.....	85
5.4.14 Importing an API Automation Case.....	89
5.4.15 Executing an API Automation Test Case.....	92
5.5 Custom Automation Test Cases.....	93
5.5.1 Creating a Custom Automation Test Case.....	93
5.5.2 Importing a Custom Automation Test Case.....	94
5.6 Advanced Configurations of API Automation.....	96
5.6.1 Built-in Functions.....	96
5.6.1.1 Binary Addition Operation.....	96
5.6.1.2 Binary Subtraction Operation.....	98
5.6.1.3 Binary Multiplication Operation.....	100
5.6.1.4 Binary Division Operation.....	101
5.6.1.5 Obtaining the Current Timestamp.....	104
5.6.1.6 Obtaining a Specified Timestamp.....	106
5.6.1.7 Converting a Date into a Timestamp.....	108
5.6.1.8 Converting a Timestamp into a Date.....	110
5.6.1.9 Timestamp Addition and Subtraction Operations.....	113
5.6.1.10 Generating Base64 Encoding.....	115
5.6.1.11 Generating SHA512 Encoding.....	116
5.6.1.12 Generating an MD5 Hash Value.....	118
5.6.1.13 Generating a Random Number in a Specified Range.....	120
5.6.1.14 Generating a Random String of a Specified Length.....	122
5.6.1.15 Generating a UUID.....	124

5.6.1.16 Obtaining an Array via Reverse Index.....	126
5.6.1.17 Obtaining the Element Values of an Array via Reverse Index.....	126
5.6.1.18 Converting Uppercase Letters into Lowercase Letters.....	127
5.6.1.19 Converting Lowercase Letters into Uppercase Letters.....	129
5.6.1.20 Concatenating Strings.....	130
5.6.1.21 Cutting Strings.....	132
5.6.2 Advanced Extraction Types.....	134
5.6.2.1 Character String Extraction.....	134
5.6.2.2 Regular Expression.....	135
5.7 Adding Test Cases in Batches.....	137
5.8 Managing Test Cases in the Features Directory.....	138
5.9 Test Cases and Requirements.....	139
5.10 Test Cases and Defects.....	141
5.11 Commenting on Test Cases.....	142
5.12 Filtering Test Cases.....	144
5.13 Customizing Columns to Be Displayed in the Test Case List.....	145
<b>6 Testing Execution.....</b>	<b>147</b>
6.1 Introduction.....	147
6.2 Manual Test Suite.....	147
6.2.1 Creating a Manual Test Suite.....	147
6.2.2 Executing a Manual Test Suite.....	148
6.3 API Automation Test Suite.....	149
6.3.1 Creating an API Automation Test Suite.....	150
6.3.2 Executing an API Automation Test Suite.....	152
6.4 Automation Custom Test Suite.....	152
6.4.1 Creating a Custom Automation Suite.....	152
6.4.2 Executing a Custom Automation Suite.....	154
<b>7 Quality Report and Assessment.....</b>	<b>156</b>
7.1 Introduction.....	156
7.2 Quality Report — Project Dashboard.....	157
7.3 TestPlan Homepage - Personal Dashboard.....	159
7.4 Custom Test Report.....	159
7.5 Evaluating Test Quality.....	161
<b>8 Settings.....</b>	<b>166</b>
8.1 Notification.....	166
8.2 User Management.....	168
8.3 Test Case Settings.....	168
8.4 Test Suite Settings.....	169
8.5 Setting a Custom Execution Type.....	169
8.6 Other Settings.....	170

---

**9 Roles and Operation Permissions.....171**

# 1 Service Entries

---

You can access CodeArts TestPlan through the [project homepage](#) or [navigation entry](#).

## Project Homepage

Access CodeArts TestPlan through the project homepage. The **Testing Plan** page is displayed by default. You can switch to **Testing Design, Testing Case, Testing Execution, Quality Report**, and **Quality Assessment** to manage test activities for the current project.

**Step 1** Log in to CodeArts.

**Step 2** Hover the cursor over the project card, click **Testing** at the bottom of the card. The **Testing Plan** page is displayed by default.


----End

## Navigation Entry

Access CodeArts TestPlan through the navigation entry. The **Testing Plan** homepage is displayed by default. The statistics of the current login user are displayed, including test plans, test cases, test suites, and defects. You can also select a project on the page to manage test activities.

**Step 1** Log in to CodeArts.

**Step 2** Access CodeArts TestPlan using either of the following methods:

- On the top navigation bar, choose **Services > TestPlan**.
- Click  in the upper left corner of the page. In the left pane, click **TestPlan**.

----End



# 2 Testing Plan

---

- [2.1 Introduction](#)
- [2.2 Creating and Editing a Test Plan](#)
- [2.3 Designing and Executing a Test Plan](#)
- [2.4 Managing and Measuring a Test Plan](#)

## 2.1 Introduction

The standard test process includes the test plan, test design, test execution, and test report.

- In the test plan and test design stages, you should specify the test scope and objectives, formulate test policy, prepare test tools and test environments, establish test models, design test cases, and develop automated testing scripts.
- A test plan specifies the test time, scope, and objectives, and manages all test activities. It can be specific to a version, iteration, or project.

This chapter describes how to manage test plans from the following aspects:

- [2.2 Creating and Editing a Test Plan](#)
- [2.3 Designing and Executing a Test Plan](#)
- [2.4 Managing and Measuring a Test Plan](#)

## 2.2 Creating and Editing a Test Plan

### Creating a Test Plan

 NOTE

Only the **Epic**, **Feature**, and **Story** work items of a Scrum project can be added to a test plan as 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** Click **Create** in the upper part of the page. The **Create Test Plan** page is displayed.

**Step 4** Set **Name**, **Version** (optional), **Processor**, **Plan Period**, **Associated Iteration** (optional), and **Description** (optional), and click **Next**.

The screenshot shows the 'Create Test Plan' form with the following fields:

- Name**: Required field, empty text input.
- Version**: Optional field, text input with placeholder 'Enter the version of the software to be tested.'
- Processor**: Required field, dropdown menu with 'apitest\_intl01' selected.
- Plan Period**: Required field, date range selector with 'select start date' and 'select end date' options.
- Associated Iteration**: Optional field, dropdown menu with 'Select an associated iteration.' placeholder.
- Description**: Optional field, text area with a note 'The maximum length is 1000 characters.'

At the bottom of the form, there are two buttons: 'Cancel' and 'Next'.

**Step 5** Select the execution mode, add a requirement, and click **Save** to complete the test plan creation.

- The execution mode selected here can be modified in the test plan later.
- After an execution mode is selected, menus corresponding to the mode are generated on the test case and test execution pages to manage the cases and suites of **Manual Test** and **Auto API Test**. Preset statistics reports of the corresponding mode are displayed in the quality report.

----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, you can modify the test plan (including the test plan name, description, execution mode, and basic information). After editing the test plan, click **Save**.
- On the **Requirements** tab page, you can add or delete requirements within the current test plan scope. The operations are the same as those in **Creating a Test Plan**.
- On the **Test Cases** tab page, you can view the test cases in the plan, and add test cases from the version to which the test plan belongs.
- On the **Operation History** tab page, you can view the editing history of the test plan.

## 2.3 Designing and Executing a Test Plan

### Designing a Test Plan

Designing a test plan is to design test cases, develop automated testing scripts, and prepare test data based on the test requirements specified in the 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** Select a test plan to be designed from the list.

- Move the cursor over **Design** to view the design progress of the test plan, including the number of test cases, total number of requirements, and number of covered requirements.

When the number of test cases is greater than 0, the circle around **Design** changes from gray to blue, indicating that the test plan is being designed, as shown in the following figure.

- Click **Design**. The **Testing Case** page is displayed.

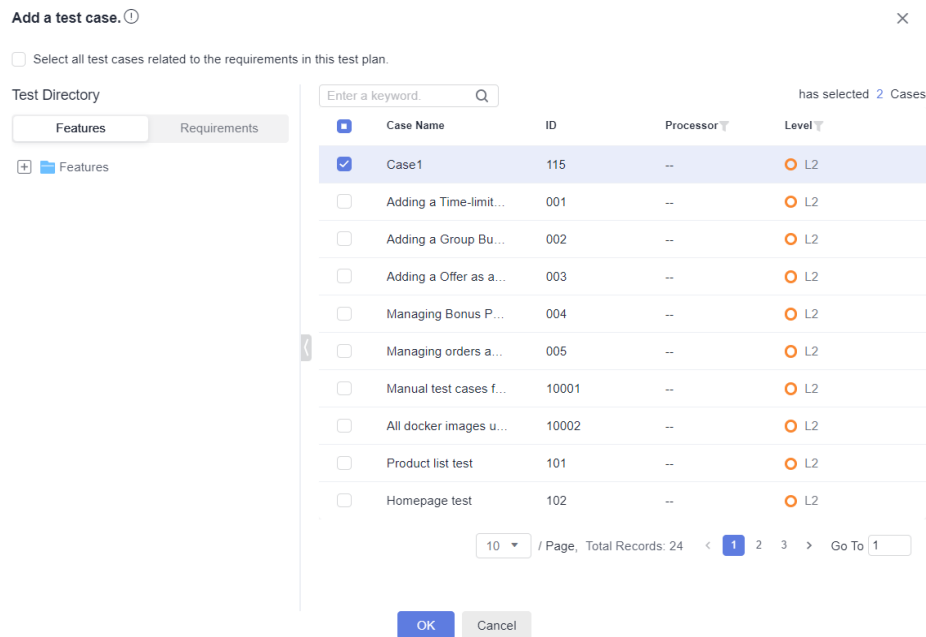


**Step 4** On the **Manual Test** tab page, click **Import** in the right area and select **Adding an Existing Case** from the drop-down list.

 **NOTE**

If no test case has been created or you need to create a test case, click **Create**. For details, see [5.3.1 Creating a Manual Test Case](#). The new test case is also added to the test case library.

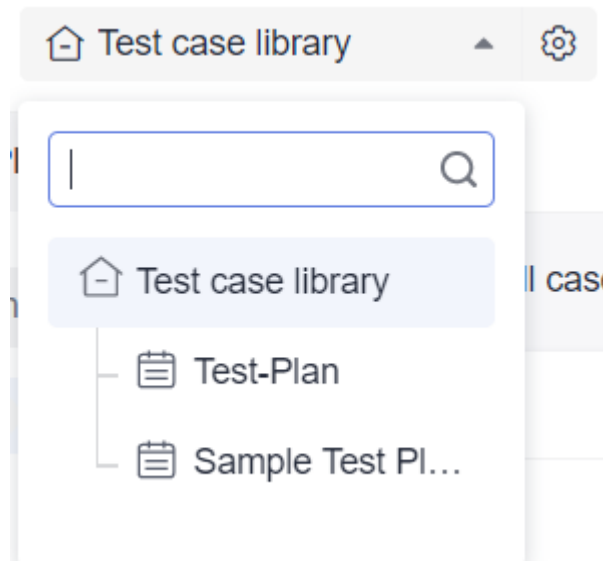
**Step 5** In the displayed dialog box, select a test case and click **OK**.



**Step 6** Click the **Auto API Test** tab to add or create API automation test cases to the test plan. For details about the method, see [Step 4](#). For details about how to create a case, see [5.4.1 Creating an API Automation Test Case](#).

**Step 7** 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 (including test cases that belong to or do not belong to the test plan) in the current version. You can maintain the global test case library as required.



----End

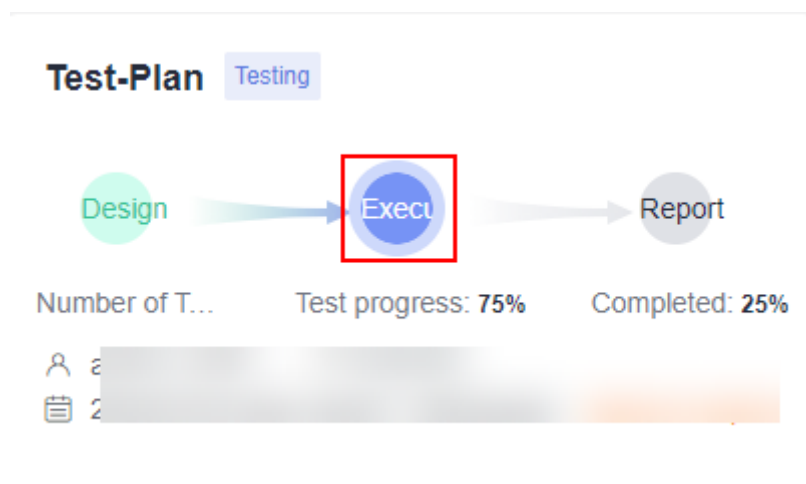
## Executing a Test Plan


**Step 1** Return to the **Testing Plan** page and select the test plan to be executed from the list.

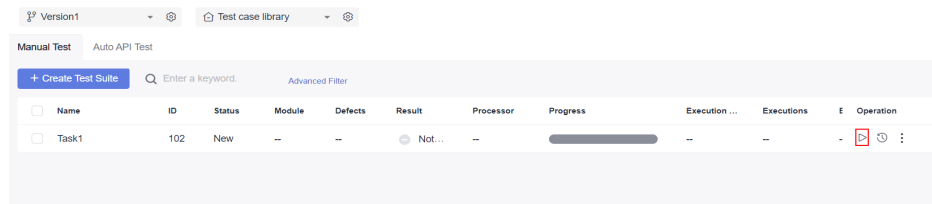
- Move the cursor over **Execute** to view the execution progress of the test plan, including the test progress, number of executed test cases, pass rate, and completed defects/total defects.

When the number of executed test cases is greater than 0, the circle around **Execute** changes from gray to blue, indicating that the test plan is being executed.

- Click **Execute**. The **Testing Execution** page is displayed.



**Step 2** On the **Manual Test** tab page, click  in the **Operation** column to execute a manual test suite of multiple test cases. If no test suite is available, create one by following instructions provided in [6.2.1 Creating a Manual Test Suite](#).



**Step 3** Click the **Auto API Test** tab and execute an API automation suite. For details, see [6.3.2 Executing an API Automation Test Suite](#).

**Step 4** After you manually set the status of all test cases in the test plan to **Completed**, the status of the test plan is automatically updated to **Completed**.

**NOTE**

On the **Testing Plan** page, click **Execute**. The **Testing Execution** page is displayed. You can create a test suite to execute the cases in the test plan.

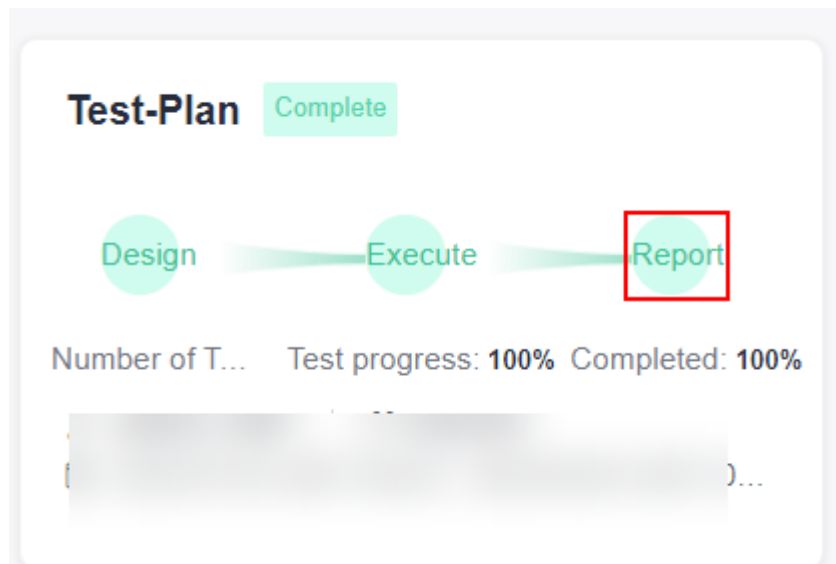
----End

## 2.4 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** Select the test plan whose report you want to view from the list and click **Report**.



**Step 4** View the quality report of the test plan.

- The page displays the current requirement coverage, defects, case pass rate, and case completion rate of the test plan, and analyzes and records test risks.
- In the **Manual Test** and **Auto API Test** areas, statistics on the execution of test cases and the number of defects are displayed by execution mode.
- To add more reports to the page, 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.

----End

# 3 Testing Design

---

[3.1 Introduction](#)

[3.2 Mind Maps](#)

[3.3 Requirement Catalog Test Design Strategies](#)

[3.4 Feature Catalog Test Design Strategies](#)

## 3.1 Introduction

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

With the help of mind mapping, heuristic test design can be performed and the design process can be visualized.

Based on different design inputs, there are two processes: requirement > scenario > test point > test case and feature > scenario > test point > test case. The final test scheme and test cases are generated.

This chapter describes how to design test cases from the following aspects:

- [3.2 Mind Maps](#)
- [3.3 Requirement Catalog Test Design Strategies](#)
- [3.4 Feature Catalog Test Design Strategies](#)

## 3.2 Mind Maps

### 3.2.1 Creating a Mind Map

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 the permission to create mind maps (that is, you have other roles except the viewer, participant, and O&M manager in the project). For details, see [User Management](#).

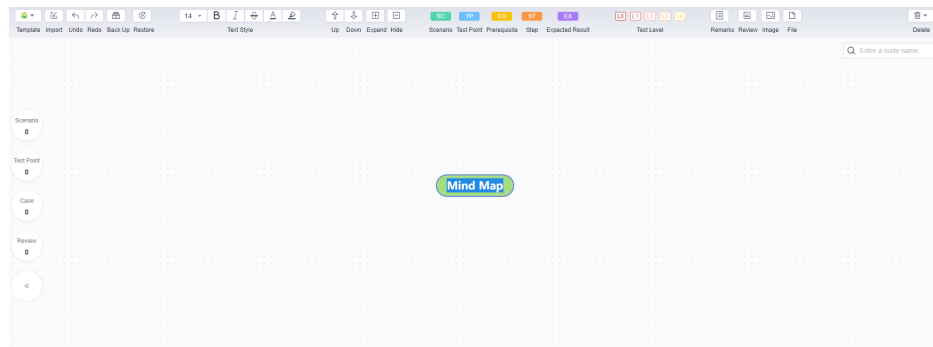
## Direct Creation


**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 **Create** in the upper left corner of the page.

The new mind map page is displayed. The root node is displayed in the middle of the page, and the name of the root node is automatically set to **Mind Map**. You can double-click the root node to change its name.



**Step 4** Click  in the upper left corner of the page. The test design list is displayed. The name of the root node of the mind map is displayed in the list.

----End

## Using a 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 **Templates** in the upper left corner of the page.


**Step 4** Select a template as required. 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 **Back** in the upper left corner of the page. The test design list is displayed.

The name of the root node of the mind map is displayed in the list.

----End

To open multiple mind maps on multiple tab pages, click  in the **Operation** column and choose **New tab opens**.

## 3.2.2 Drawing a Mind Map

After the operation is complete in [3.2.1 Creating a Mind Map](#), click the name of the mind map to be edited on the **Testing Design** page to draw the mind map.

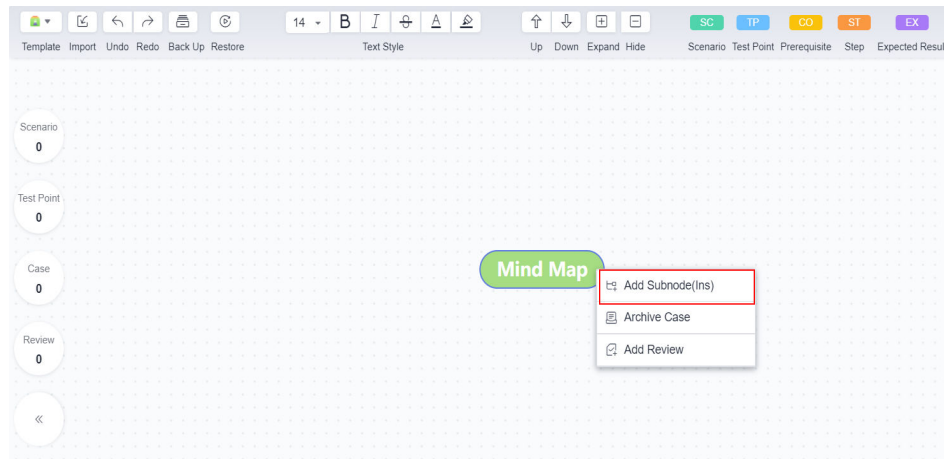
### Adding a Node

You can 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 the created mind map, select a node, and add a subnode in either of the following ways:

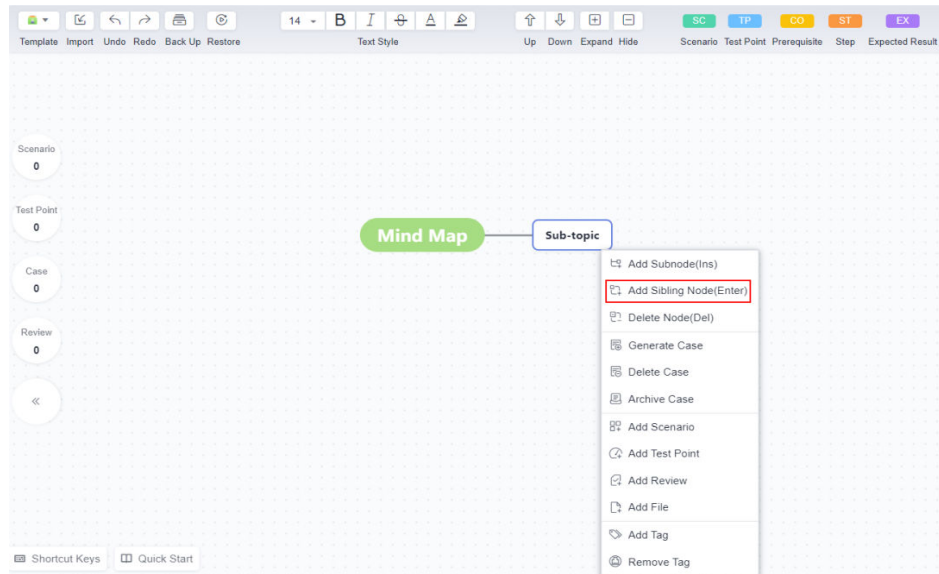
- Right-click and choose **Add Subnode (Ins)** from the shortcut menu.
- Press **Insert** or **Tab**.



- Add a Sibling Node

Access the created mind map, select any node except the root node, and add a sibling node in 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 the created mind map, select any node except the root node, and delete the node in either 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**.

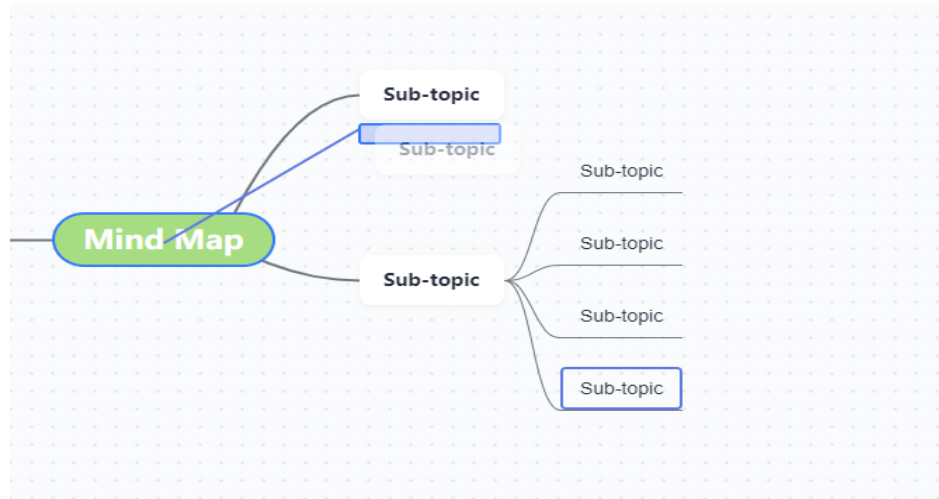
## Moving a Node

- Move Up/Down

Access the created 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 between nodes of the same level.

- Dragging a Node


Access the created mind map, select the node to be moved, click the node, drag the node to the required position, and release the left mouse button.



















### 3.2.3 Managing a Mind Map

#### Performing Basic Operations


After entering the mind map, you can perform the following operations in addition to editing nodes.

Operation	Description
Delete a mind map	In the upper right corner of the mind map page, click  , select <b>Delete Mind Map</b> , and click <b>Confirm</b> to delete the current mind map. The deleted mind map is moved to the recycle bin.

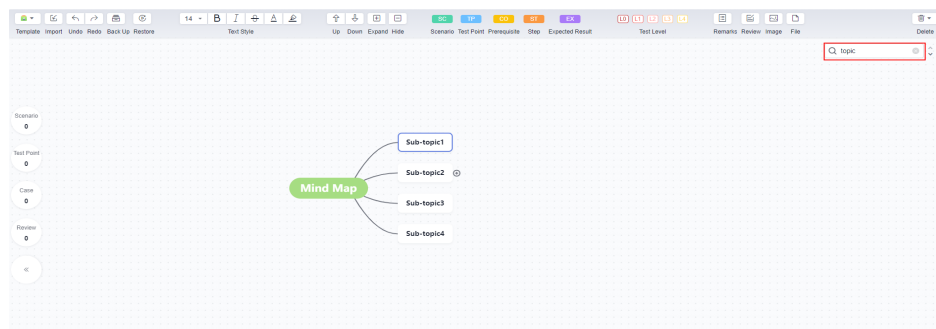
Operation	Description
Recycle bin	<p>Click  in the upper right corner of the mind map and select <b>Recycle Bin</b> from the drop-down list. In the dialog box that is displayed, view the list of deleted mind maps.</p> <p>For mind maps in the recycle bin, you can perform the following operations:</p> <ul style="list-style-type: none"> <li>Click  and <b>Confirm</b> to restore the mind map in the corresponding row.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>Restoring the mind map will replace the content on the current page. Exercise caution when performing this operation.</li> <li>Once a record in the recycle bin is restored, the record is removed from the list.</li> <li>Restoring the recycle bin will overwrite the current mind map. You are advised to create a blank mind map to restore the recycle bin.</li> </ul> <ul style="list-style-type: none"> <li>Click  to view details about the mind map in the corresponding row.</li> <li>Click  and <b>OK</b> to permanently delete the mind map in the corresponding row. Deleted mind maps cannot be restored. Exercise caution when performing this operation.</li> </ul>
Import	<p>Click  on the toolbar above the mind map, and then click <b>Confirm</b>. In the dialog box that is displayed, select an existing local <b>.xmind</b> file to import the file to the current page.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>A file whose size does not exceed 20 MB can be imported.</li> <li>A maximum of 100 <b>.xmind</b> files in a compressed file can be imported.</li> <li>The imported content will replace the mind map on the current page. Exercise caution when performing this operation.</li> <li>After the mind map is imported successfully, the root node is not changed.</li> </ul>
Backup	<p>Click  on the toolbar above the mind map. In the dialog box that is displayed, enter the name and description, and click <b>Confirm</b> to create a backup for the mind map on the page.</p>
Restore	<p>Click  on the toolbar of the mind map. In the dialog box that is displayed, select a backup and click . In the dialog box that is displayed, click <b>Confirm</b> to restore the backup mind map to the current page.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>Restoring the mind map will replace the content on the current page. Exercise caution when performing this operation.</li> <li>Before the mind map is restored, the content of the current mind map is automatically backed up.</li> </ul>

Operation	Description
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  .
Hide All	Select a node with  and click  on the toolbar above the mind map to hide all subnodes under the node. The icon next to the node changes to  .

## Searching for Nodes

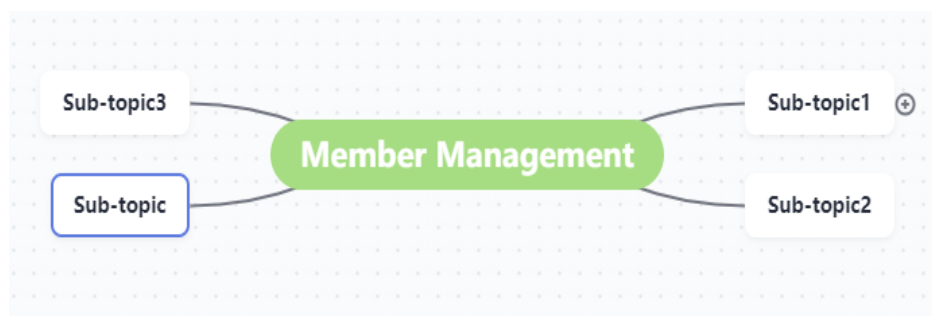
Enter a keyword in the search box in the upper right corner of the mind map and click  to find the node containing the keyword in the mind map. The found node is marked in a blue box.

If multiple search results are matched, you can view the number of matched items on the right of the search box and the sequence number of the matched item of the current node. You can click the up or down arrow to adjust the matching items.




## 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.

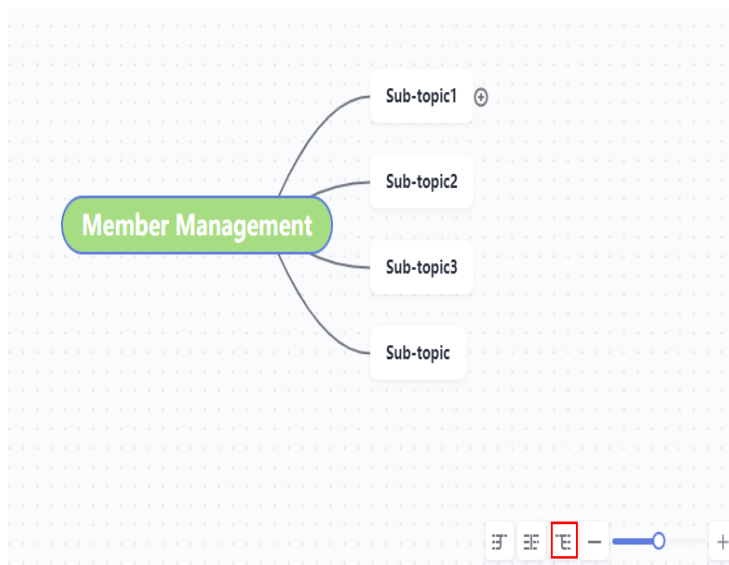





Click the toolbar in the lower right corner of the mind map to adjust the layout of the mind map.

- 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

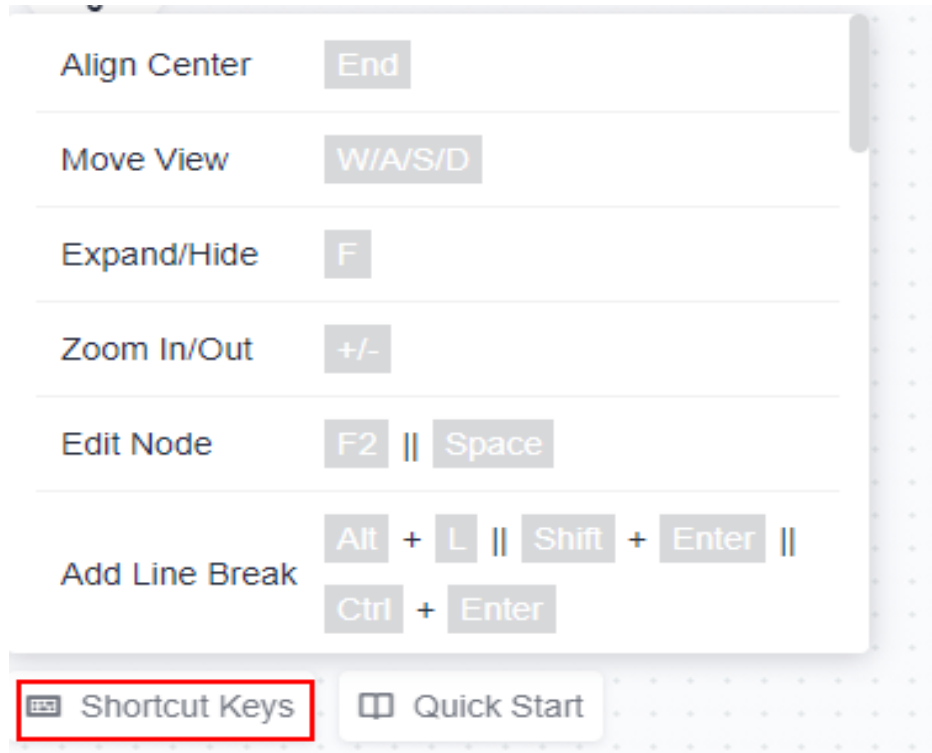
You can use the following shortcut keys to edit the mind map during test design.

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

Operation	Shortcut Key
Expand/Hide	F
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
Add 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 Case Level	Ctrl+0 / 1 / 2 / 3 / 4
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.



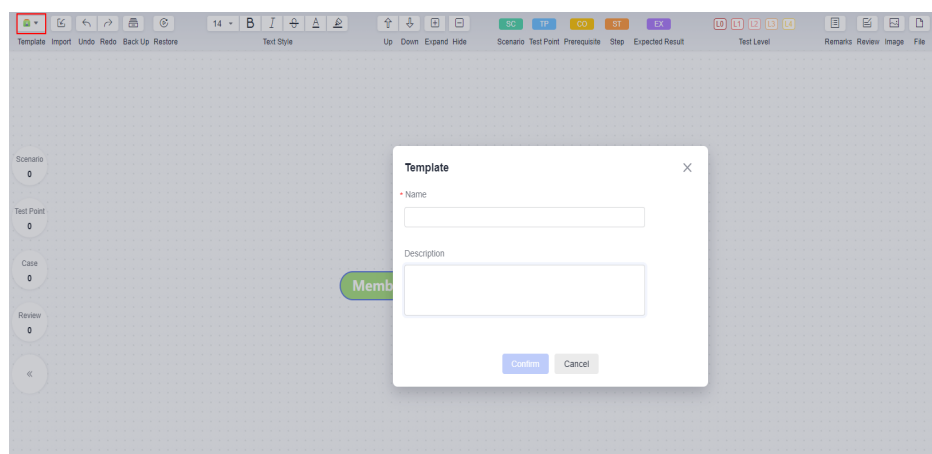


### 3.2.4 Managing Templates

#### Save As

A created mind map can be saved as a template.

- Step 1** Access the created 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 the dialog box that is displayed, enter a name and click **Confirm**.




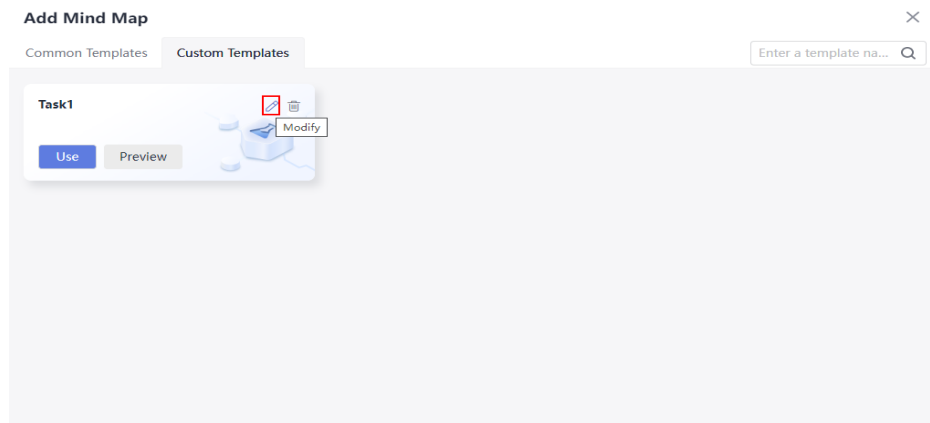
- Step 4** Click the back icon in the upper left corner to return to the test design list. Click **Templates** in the upper left corner of the page, and click the **Custom Templates** tab. The saved template is displayed in the dialog box.

----End

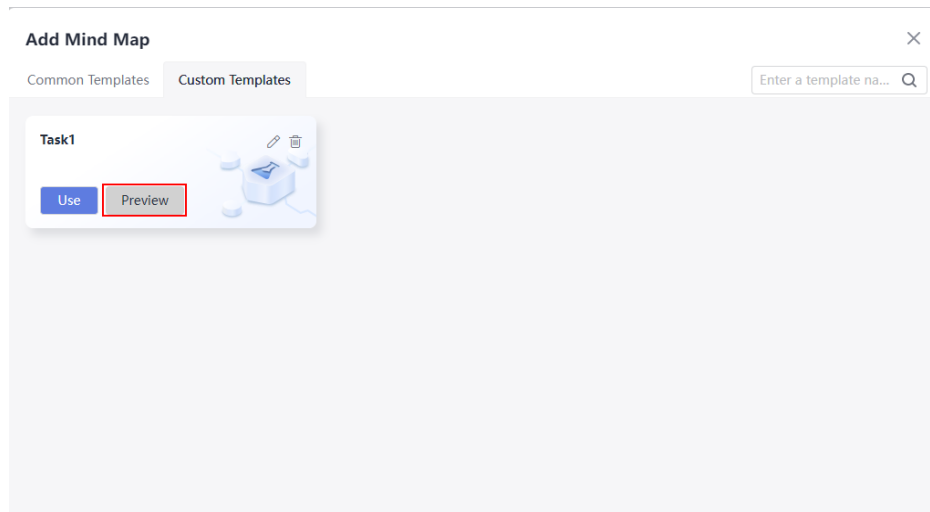
## Editing a Template

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  corresponding to the template.



- Step 3** Edit the template as required. After the editing is complete, click **Back** 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 dialog box that is displayed, locate the edited template and click **Preview** to view the modified template details.



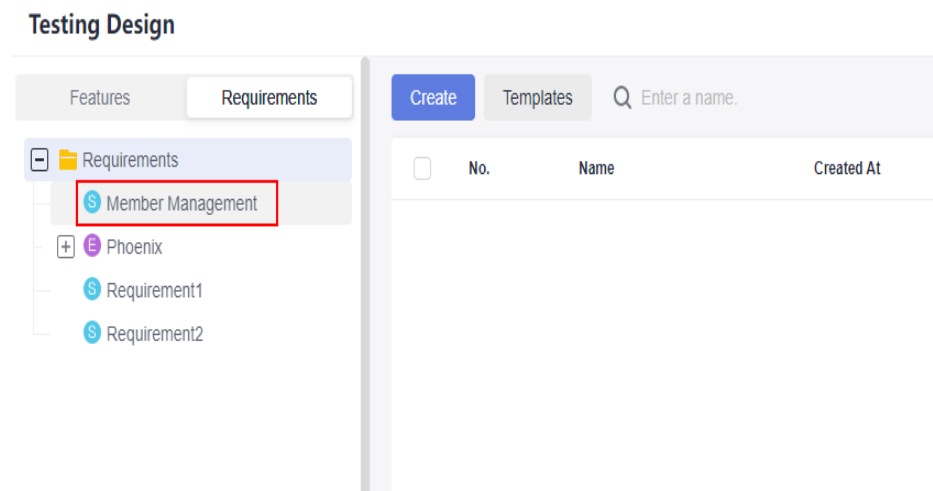
----End

## 3.3 Requirement Catalog Test Design Strategies

### 3.3.1 Creating a Mind Map Based on Requirements

CodeArts TestPlan allows you to create a mind map for a requirement, break down test scenarios, analyze test points, and output test schemes and test cases based on the requirement.


- 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 **Requirements** on the left, select a requirement, and click **Create** or **Templates** in the upper left corner.




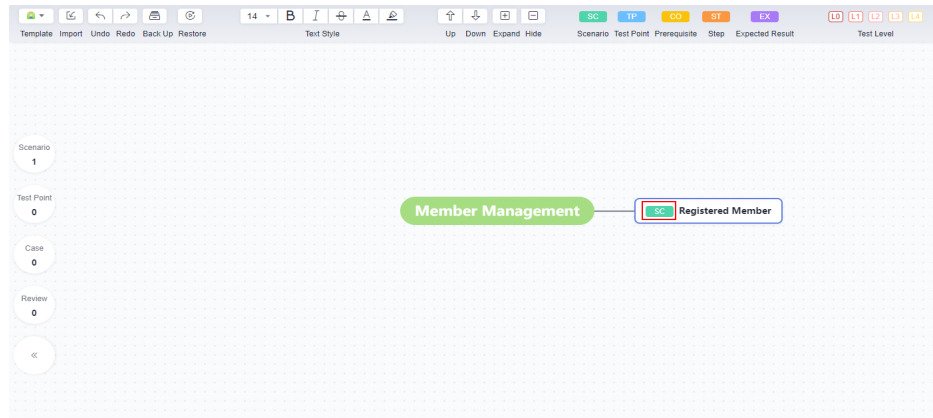
----End

### 3.3.2 Designing a Scenario

After the operation in [3.3.1 Creating a Mind Map Based on Requirements](#) is complete, you can design a scenario based on this requirement.

- Step 1** Go to the created mind map.
- Step 2** Add a subnode to the root node. For details, see [Adding a Node](#).
- Step 3** Select the node created in [Step 2](#) and click  on the toolbar above the mind map.

If  is displayed next to the selected node, the scenario is added successfully.




----End


### 3.3.3 Designing a Test Point

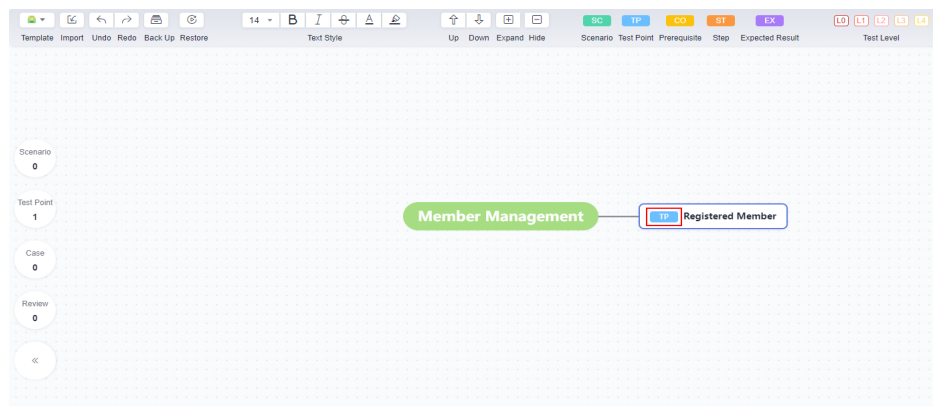
After the operation in [3.3.1 Creating a Mind Map Based on Requirements](#) is complete, you can design a test point based on this requirement.

**Step 1** Go to the created mind map.

**Step 2** Add a subnode to the root node. For details, see [Adding a Node](#).

**Step 3** Select the node created in [Step 2](#) and click  on the toolbar above the mind map.

If  appears in front of the selected node, the test point is added successfully.



----End

#### NOTICE

The test point name can contain only 3 to 128 characters and allows uppercase letters, lowercase letters, digits, and special characters (-\_/\*&`'^~;(){}=+,x...—!@#\$%.[<>?-"').

### 3.3.4 Designing a Test Case

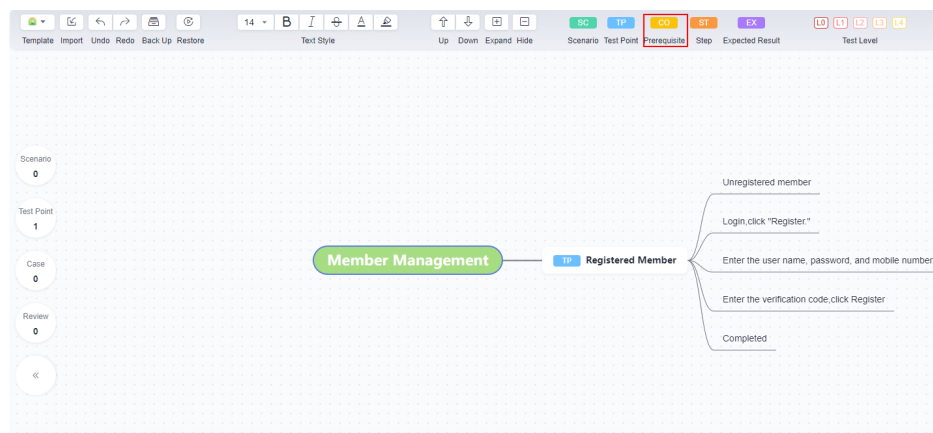
After the operation in [3.3.3 Designing a Test Point](#) is complete, the test cases can be further refined.

A test case consists of three parts: prerequisites, step, and expected result. Set the three parts during test case design.

**Step 1** Go to the created mind map.

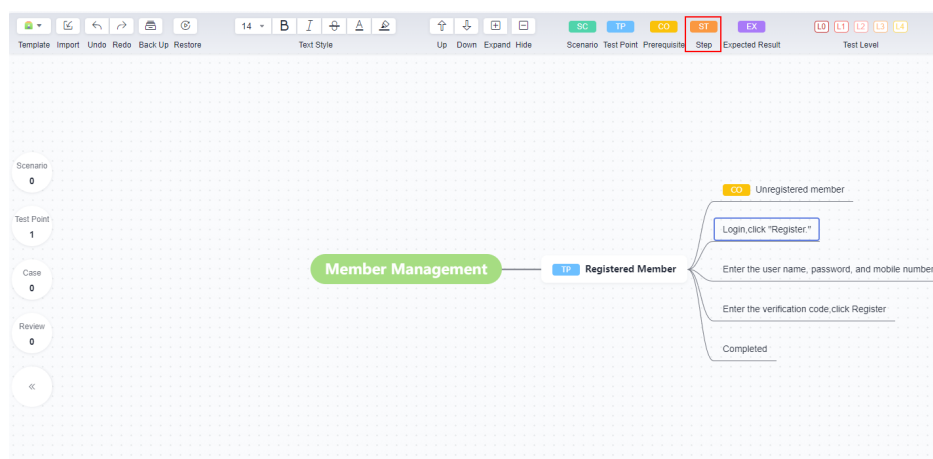
**Step 2** Under the registered member node, create subnodes as required (prerequisites, step, and expected result). For details, see [Adding a Node](#).

**Step 3** Select the node as the prerequisite and click **CO** on the toolbar above the mind map.



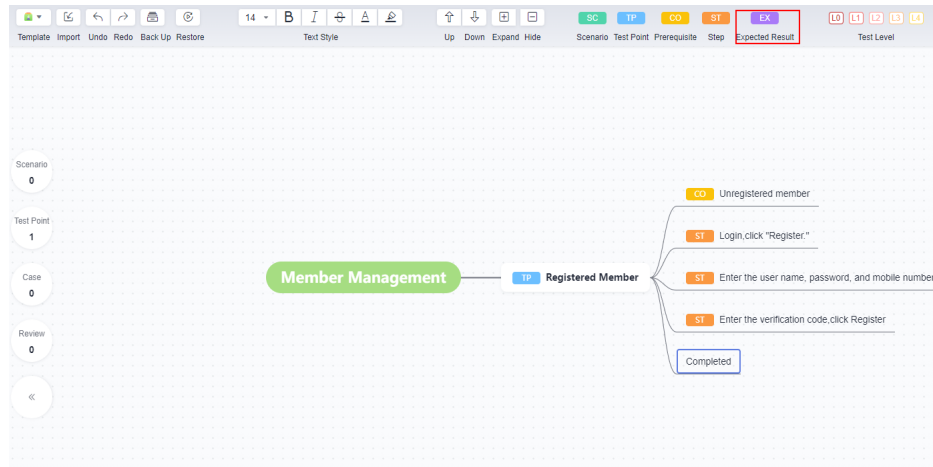
If **CO** is displayed in the node, the setting is successful.

**Step 4** Select the node that is used as the step and click **ST** on the toolbar above the mind map.



If **ST** is displayed in the node, the setting is successful.

**Step 5** Select the node as the expected result and click **EX** on the toolbar above the mind map.



If **EX** is displayed in the node, the setting is successful.

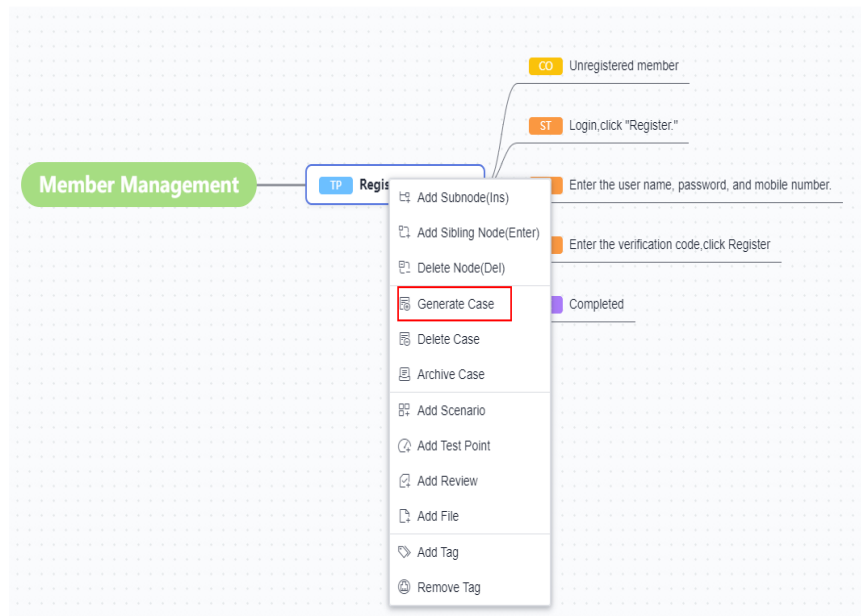
----End


### 3.3.5 Generating a Test Case

After the operation in [3.3.4 Designing a Test Case](#) is complete, you can use the mind map to generate a real test case.

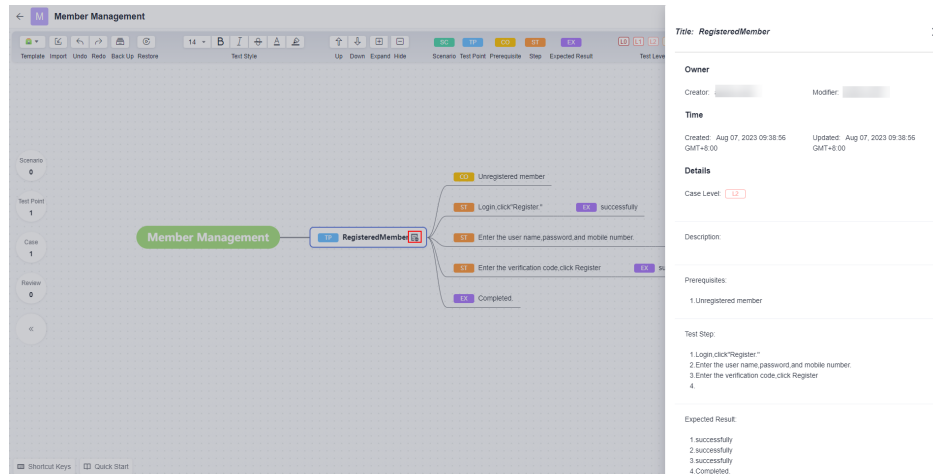
#### Generating a Test Case

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



**Step 2** If  is displayed on the node, the operation is successful. In this case, a draft case is generated.

Click . The case details are displayed on the right of the page.

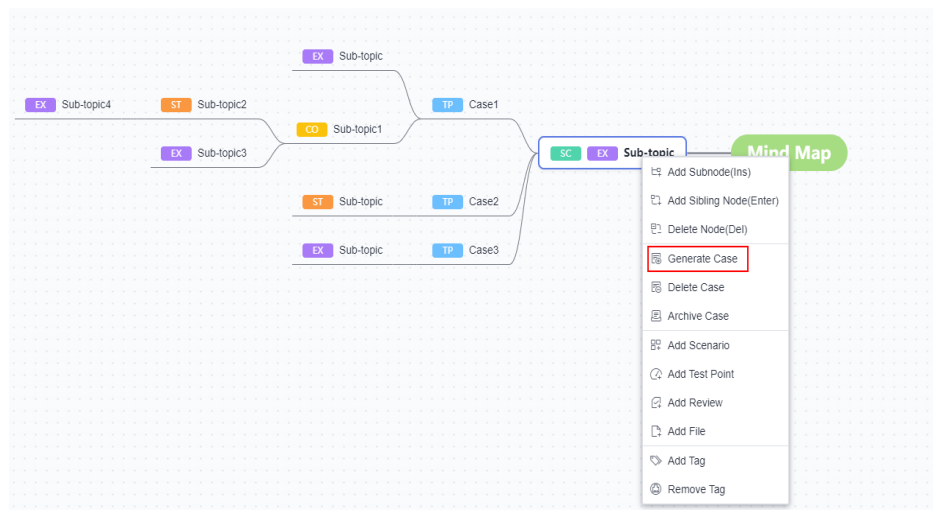



----End

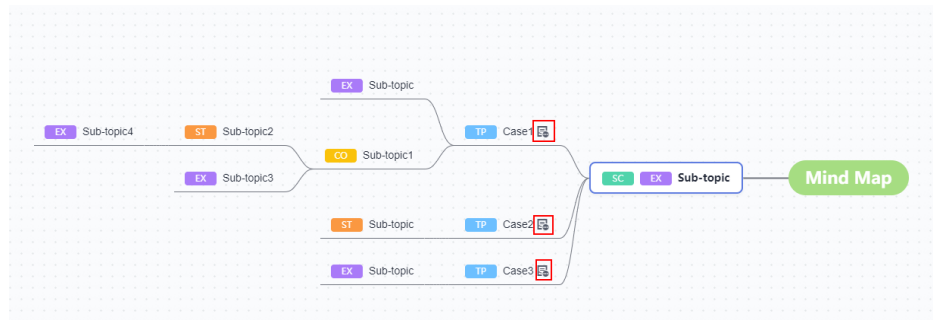
## Generating Cases in Batches

If multiple test points are set in a scenario, draft cases can be generated in batches based on the scenario.

In the mind map, select a scenario that contains multiple test points, right-click the node for which a case is to be generated and choose **Generate Case**.



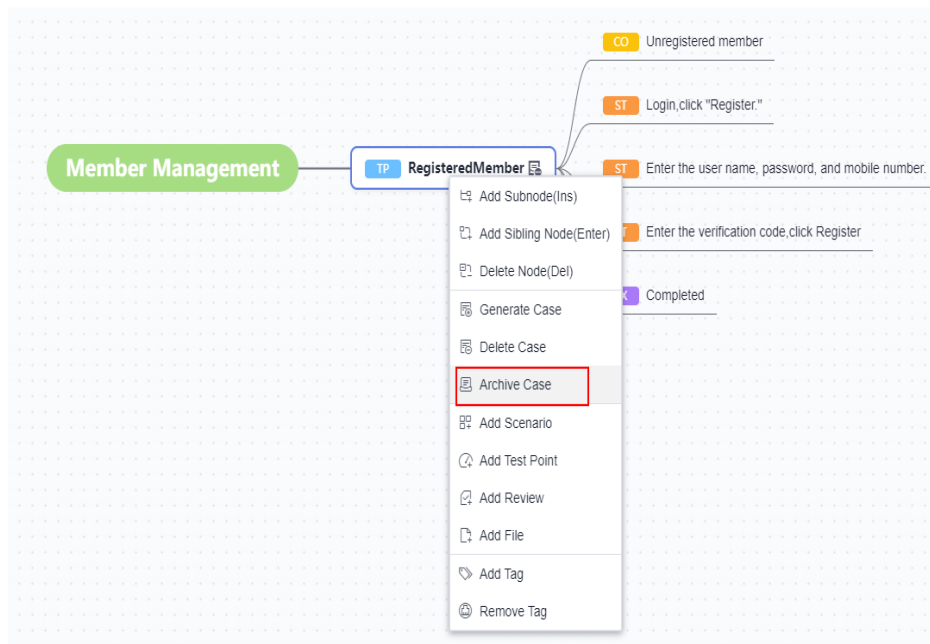
If  is displayed on all test point nodes in this scenario, the operation is successful, and draft test cases are generated.



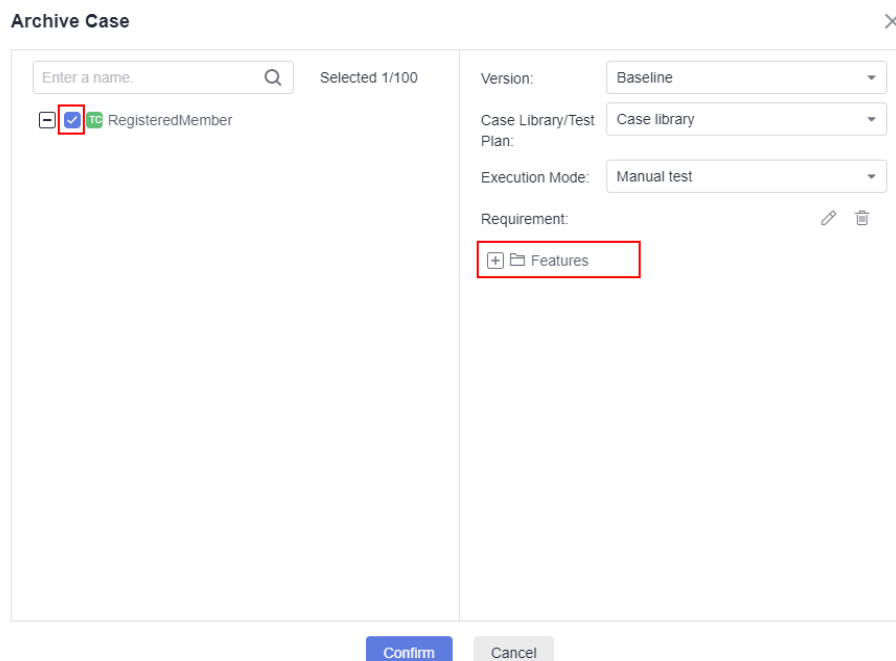
## Archiving Cases

After the generated test cases are archived, real test cases are generated. You can find the test case records on the test case page.


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




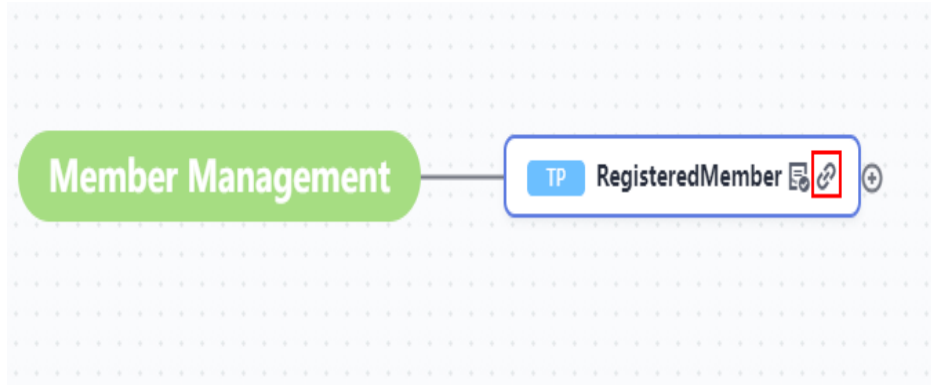
- Step 2** Select the cases to be archived on the left. On the right of the page, set **Version**, **Case Library/Test Plan** where the cases are to be stored, and **Execution Mode**, select the feature directory, and click **Confirm**.





**Step 3** If  is displayed in the node, the operation is successful. You can find the test case on the **Testing Case** page.

Click . The test case details page is displayed.



----End

### 3.3.6 Adding a Tag and Remarks

During test design, you can add tags and remarks for nodes as auxiliary tools.

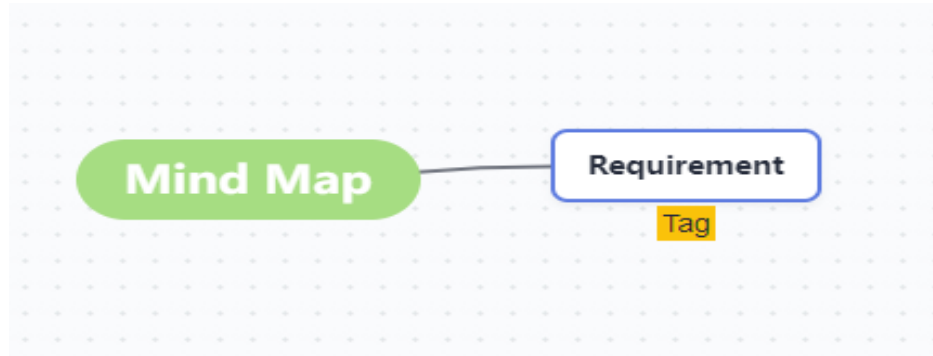
#### Adding a Tag

**Step 1** Access the mind map, right-click a node, and choose **Add Tag**.




**Step 2** Enter the tag content in the text box under the node as required and click any position on the screen to save the tag content.

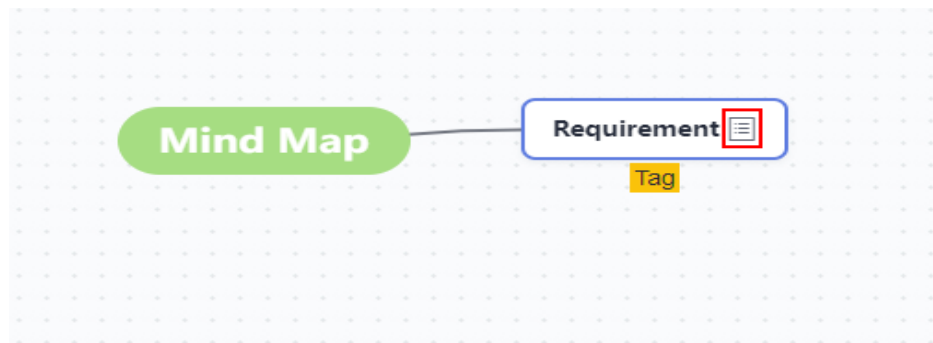
**Step 3** After the settings are saved, a tag with a yellow background is displayed under the node.



----End

## Adding Remarks

- Step 1** Access the 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 the remarks.
- Step 3** After the settings are saved,  is displayed in the node.

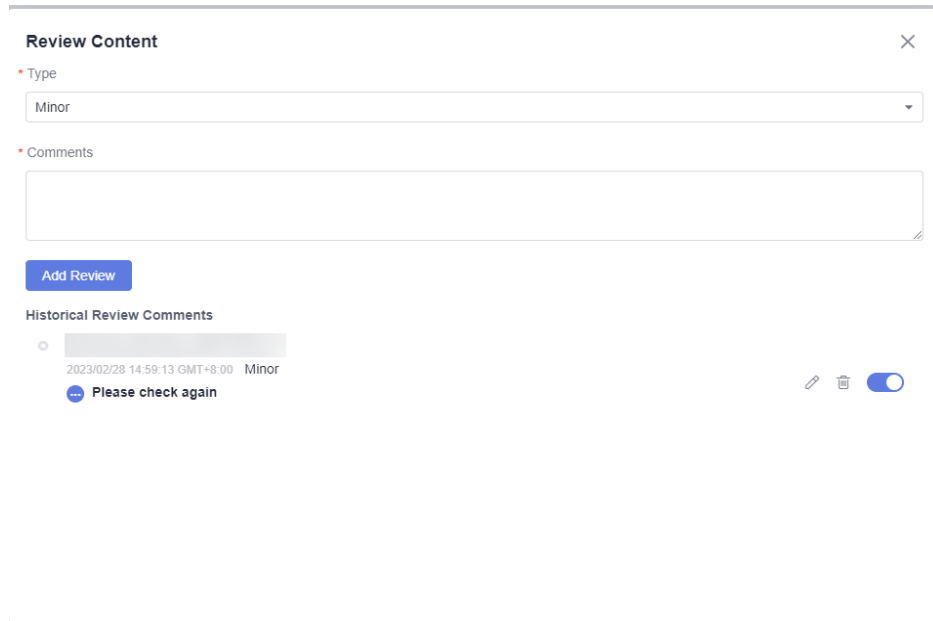



----End

## 3.3.7 Performing Online Review

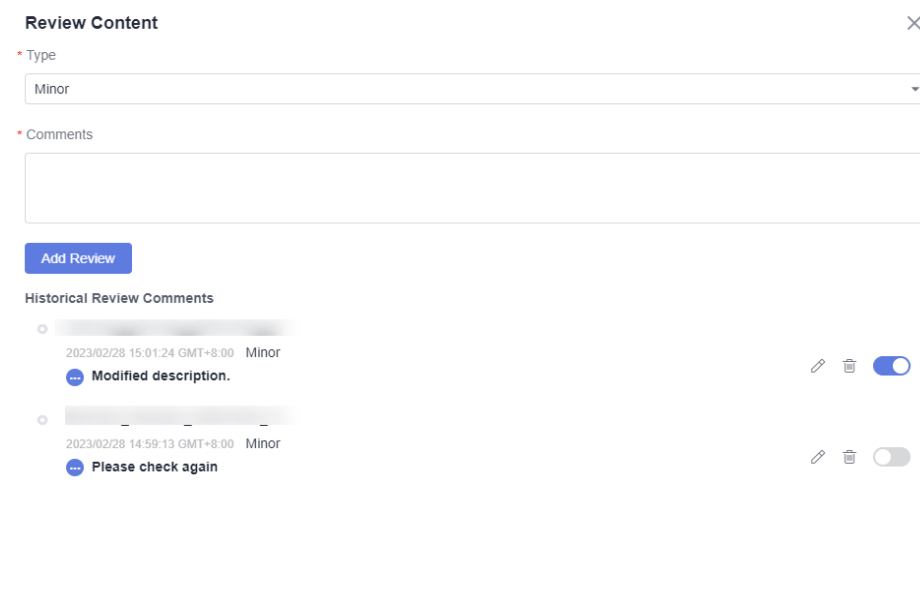
After the case design process is complete, you can review the case online and add review comments.

- Step 1** Access the mind map to be reviewed, right-click a case node, and choose **Add Review**.
- Step 2** In the dialog box that is displayed, select a type, enter comments, and click **Add Review**. The review record is added.
- Step 3** View the review record in **Historical Review Comments**.



**Step 4** Click  to modify the review comment and its type.

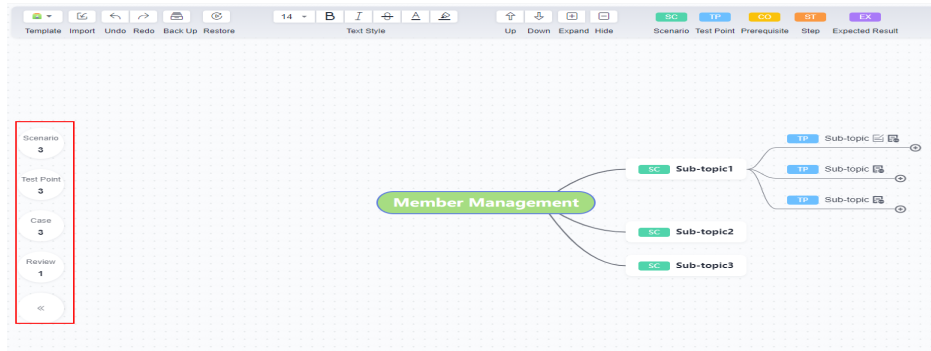
You can click the switch button on the right of a review comment to change the review comment status.



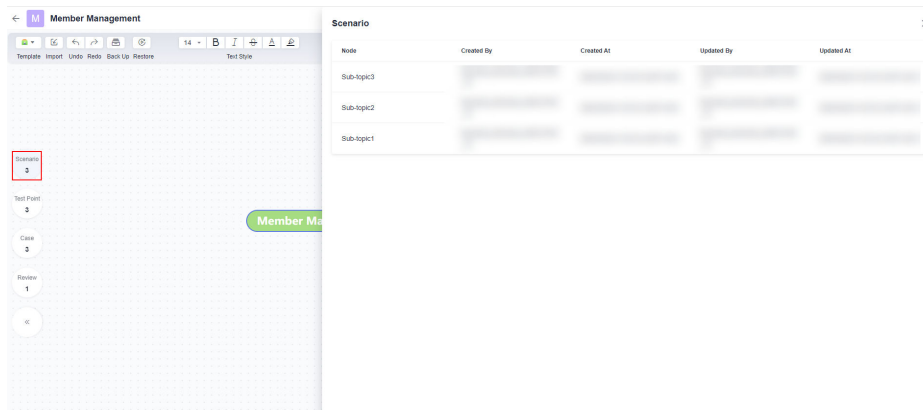
----End

### 3.3.8 Collecting Node Statistics

Access a mind map. The statistics of each type of node in the mind map are displayed on the left of the page, including the total number of scenarios, test points, cases, and reviews.



Click any statistical value. The node list corresponding to the value is displayed on the right of the page.



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

## 3.4 Feature Catalog Test Design Strategies

### 3.4.1 Creating a Mind Map Based on Features


CodeArts TestPlan allows you to create a mind map for a feature, break down test scenarios, analyze test points, and output test schemes and test cases based on the feature.

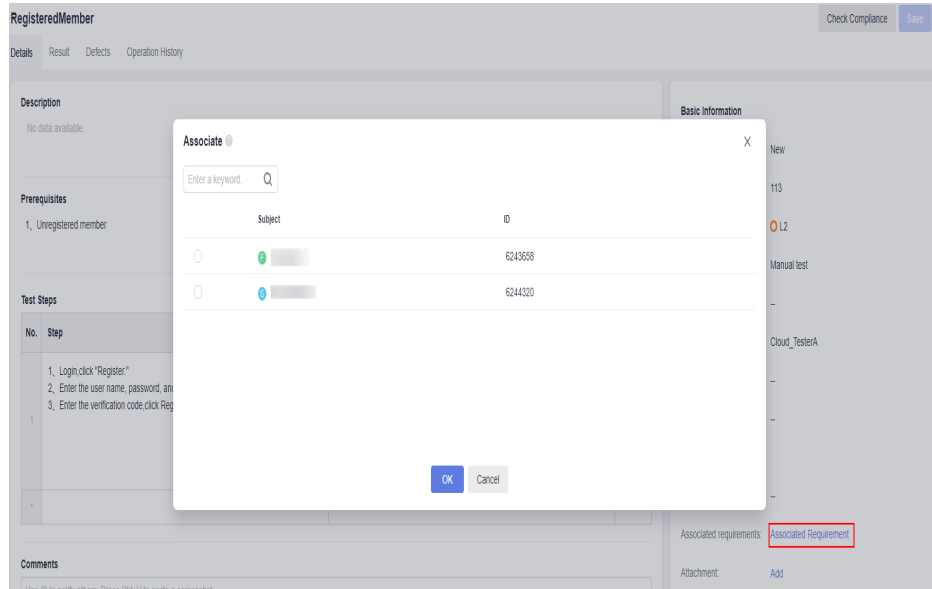
- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Design**.
- Step 3** Click **Features** on the left, select a feature directory on the left and choose **Create** or **Templates** in the upper left corner.

----End

### 3.4.2 Associated with a Requirement

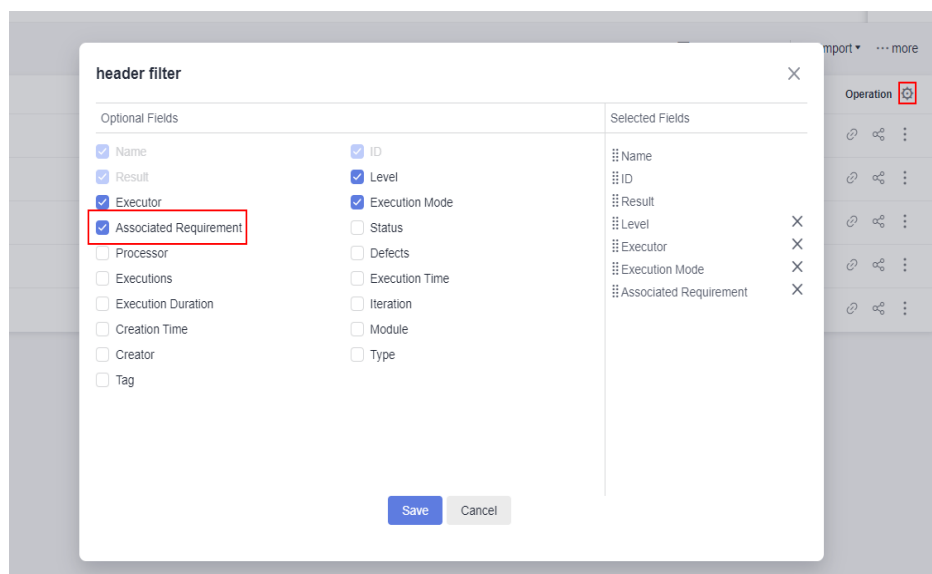
Test cases designed and archived through the feature catalog are not associated with requirements. You can perform the following steps to associate with a requirement:

- Step 1** Go to the mind map generated based on the feature directory.
- Step 2** Find the case that has been archived and needs to be associated with a requirement, and click .
- Step 3** On the right of the page, click **Associate Requirement**. In the dialog box that is displayed, select the requirement, click **OK**, and click **Save** in the upper right corner.



- Step 4** In the navigation tree, choose **Testing > Testing Case**. On the page that is displayed, find the 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 to customize column settings, select **Associated Requirement**, and click **Confirm**.



----End

# 4 Testing Management

---

## [4.1 Introduction](#)

## [4.2 Creating, Editing, and Deleting a Version](#)

## [4.3 Managing a Version](#)

## [4.4 Measuring a Version](#)

## 4.1 Introduction

A version records information, such as test cases, test case execution, test suites, test suite execution, and quality reports, about a specific software version.

- **Version:** A version records information, such as test cases, test case execution, test suites, test suite execution, and quality reports, about a specific software version.
- **Relationship between versions and test plans:** A version contains one case library and multiple test plans.
- **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. In case management, cases can be merged from a version to a baseline, cases can be cloned from a baseline to a version, and conflict management of case merging between versions is supported. This meets the requirements of concurrent tests of multiple versions and collaborative tests of multiple persons, solving the problems of test asset accumulation and reuse.

This section describes how to manage versions from the following aspects:


### [4.2 Creating, Editing, and Deleting a Version](#)

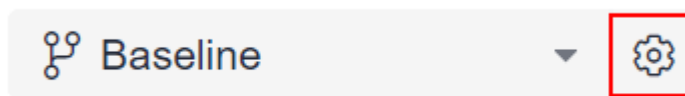
### [4.3 Managing a Version](#)

### [4.4 Measuring a Version](#)

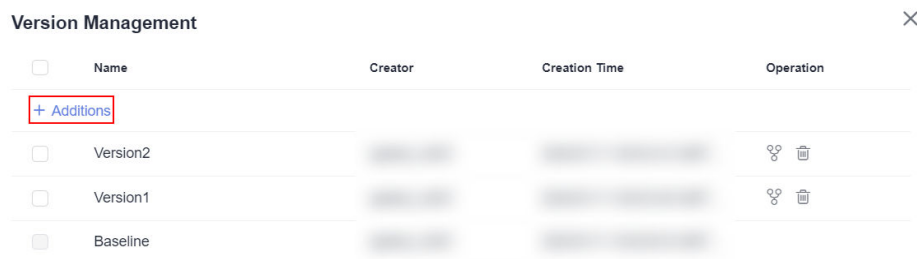
## 4.2 Creating, Editing, and Deleting a Version

### Creating a Version

- 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  on the right of **Baseline**. The **Version Management** page is displayed.




- Step 4** On the **Version Management** page, click **+ Additions**.

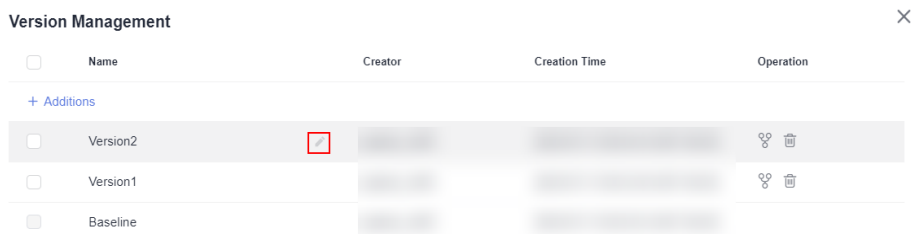


- Step 5** Enter the version name and click **Save**.


----End

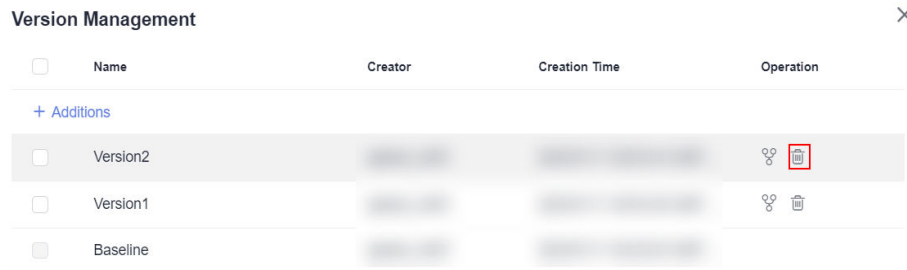
### Editing a Version

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



## Deleting a Version

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

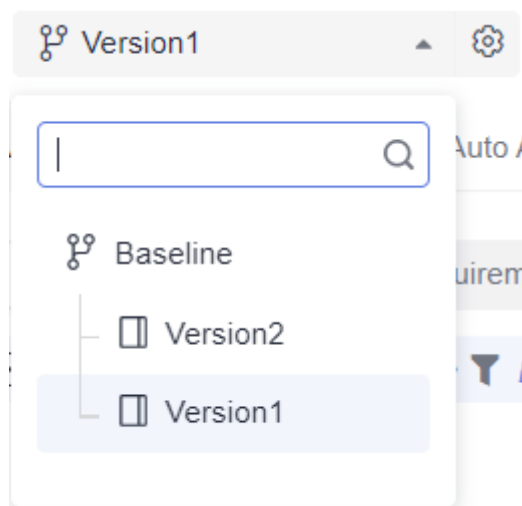


## 4.3 Managing a Version

Version management is a process of importing test cases to a baseline to a version or merging test cases from a version to a baseline based on test policies.

### Importing Manual Test Cases from Other Versions

- 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** 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, click **Import** and select **Import from Version** from the drop-down list.

#### NOTE

If no test case has been created or you need to create a test case, click **Create** to create a test case. For details, see [5.3.1 Creating a Manual Test Case](#).



**Step 5** In the displayed dialog box, select the case to be imported, select the overwriting rule, and click **OK**.

Move the cursor over the check box on the left of **Case Name** and select **Select Current Page** or **Select All Pages** to import the cases on the current page or all cases of the corresponding version to the current version.

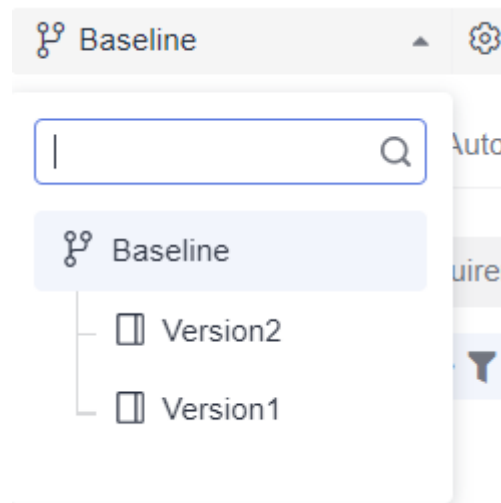
----End

## Merging Manual Test Cases in the Version into the Baseline (Testing Case Page)

**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** Select a version from the 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, click **Merge into Baseline**.

### NOTE


If no test case has been created or you need to create a test case, click **Create** to create a test case. For details, see [5.3.1 Creating a Manual Test Case](#).

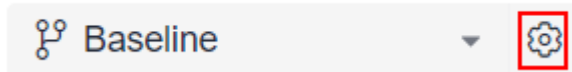
**Step 5** In the displayed dialog box, select the test case to be merged, select the overwriting rule, and click **OK**.


Move the cursor over the check box on the left of **Case Name** and select **Select Current Page** or **Select All Pages** to merge the cases on the current page or all manual test cases of the current version into the baseline.





----End

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

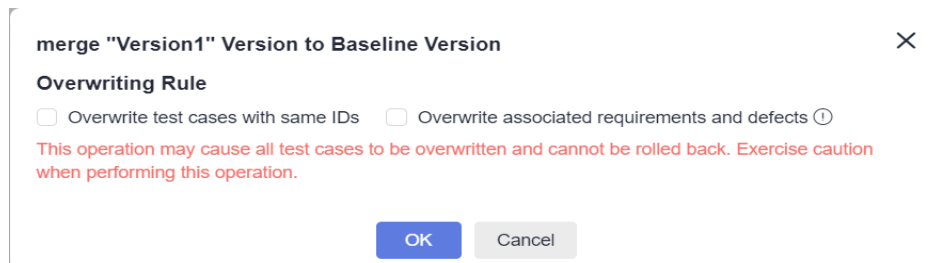
- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** Click  on the right of **Baseline**. The **Version Management** page is displayed.



- Step 4** On the **Version Management** page, select the version whose cases you want to merge into the baseline, and click  in the **Operation** column.

<input type="checkbox"/>	Name	Creator	Creation Time	Operation
<input type="checkbox"/>	Version2	e		 
<input type="checkbox"/>	Version1	e		 
<input type="checkbox"/>	Baseline	e		

- Step 5** In the displayed dialog box, select a rule and click **OK** to merge test cases.

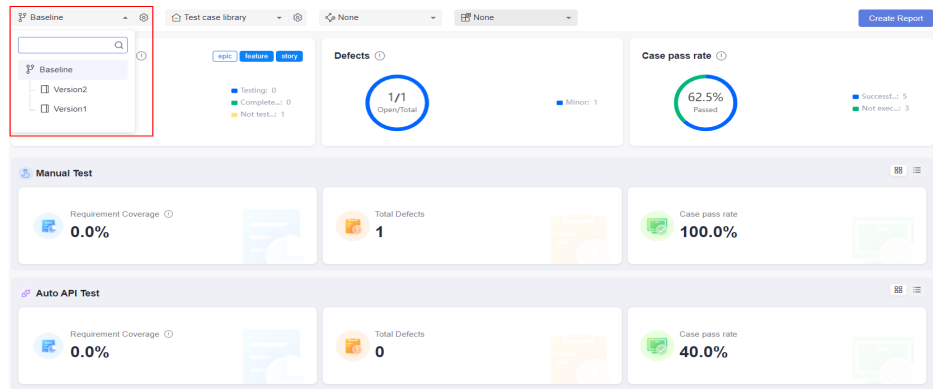


----End

## 4.4 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** View the test dashboard of the version.

You can switch the version by selecting a version from the drop-down list in the upper left corner of the page. Then, you can switch the test plan by selecting a test plan from the drop-down list and view the quality dashboard.



----End

# 5 Testing Case

---

- [5.1 Introduction](#)
- [5.2 Test Case Library](#)
- [5.3 Manual Test Cases](#)
- [5.4 API Automation Test Cases](#)
- [5.5 Custom Automation Test Cases](#)
- [5.6 Advanced Configurations of API Automation](#)
- [5.7 Adding Test Cases in Batches](#)
- [5.8 Managing Test Cases in the Features Directory](#)
- [5.9 Test Cases and Requirements](#)
- [5.10 Test Cases and Defects](#)
- [5.11 Commenting on Test Cases](#)
- [5.12 Filtering Test Cases](#)
- [5.13 Customizing Columns to Be Displayed in the Test Case List](#)

## 5.1 Introduction

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.

This chapter describes how to design test cases from the following aspects:

- [5.2 Test Case Library](#)
- [5.3 Manual Test Cases](#)
- [5.4 API Automation Test Cases](#)
- [5.6.1 Built-in Functions](#)
- [5.7 Adding Test Cases in Batches](#)

- [5.8 Managing Test Cases in the Features Directory](#)
- [5.9 Test Cases and Requirements](#)
- [5.10 Test Cases and Defects](#)
- [5.11 Commenting on Test Cases](#)
- [5.12 Filtering Test Cases](#)
- [5.13 Customizing Columns to Be Displayed in the Test Case List](#)

## 5.2 Test Case Library

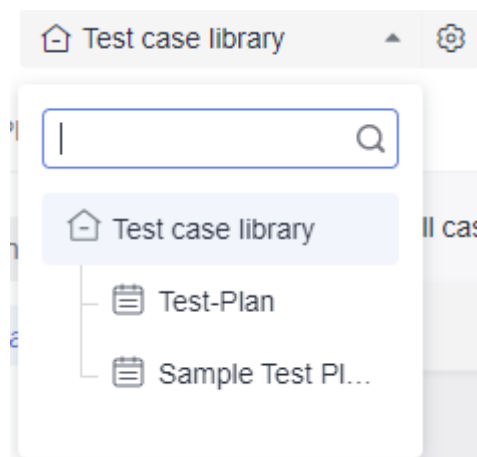
The test case library is a collection of all test cases in the current version, and allows you to view, manage, and use them.

- You can add test cases in 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** 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**. The **Test case library** is displayed by default.

You can select a test plan from the drop-down list of the test case library to view the test case details in the test plan.



----End

## 5.3 Manual Test Cases

## 5.3.1 Creating a Manual Test Case

### Prerequisites

You have the permission to create test cases (that is, you have other system roles except the O&M manager, viewer, and participant in the project).

### Procedure

Manual test cases are used to manage test scenarios and steps. You can view manual test cases on the **Testing Case > Manual Test** page.

- 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** **Test case library** and **Baseline** are displayed by default. On the **Manual Test** tab page, click **Create** on the left.
- Step 4** Enter the name, description, prerequisites, and test steps as required, and click **Save**.

The main configuration items of the manual test cases are as follows.

Configur ation Item	Description
Name	Name of a test case (mandatory). You can describe its test scenario or function.
Descriptio n	Brief description of a test case. You can 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.
Test Steps	Step description and expected result.
Tag	Tag for the current test case as required. Each test case can be associated with a maximum of 10 tags, which should be separated by spaces.
ID	Test case ID, which can be customized. If this item is left blank, a test case ID is automatically generated.

Configuration Item	Description
Case Level	<p>Test case level based on the importance of the scenario or function. The default level is <b>L2</b>.</p> <ul style="list-style-type: none"><li>• <b>L0</b>: verification of underlying functions. Each module should have 10 to 20 test cases. L0 test cases account for 5% of all test cases.</li><li>• <b>L1</b>: verification of basic functions for inherited features or before iteration acceptance. L1 test cases account for 20% of all test cases.</li><li>• <b>L2</b>: verification of important features for manual tests in non-regression versions. L2 test cases account for 60% of all test cases.</li><li>• <b>L3</b>: 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.</li><li>• <b>L4</b>: verification of special input, scenarios, and threshold conditions. L4 test cases account for less than 5% of all cases.</li></ul>
Module	Module of the current test case. The module list comes from the project settings. For details, see Setting Modules.
Version	Version number of the current test case.
Iteration	Iteration to test the current test case.
Processor	Person who needs to complete the test task.
Associated Requirement	<p>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 <b>CodeArts Req</b> service, click <b>Associate</b> to associate the test case with the requirement work item.</p> <p><b>NOTE</b> Test cases can be associated only with <b>Epic</b>, <b>Feature</b>, and <b>Story</b> work items of a Scrum project.</p>
Folder	Associated feature directory. The test case is in the <b>Other</b> directory by default.
Attachment	Attachment related to the current test case. You can associate an existing file from the <b>Files</b> page or upload an attachment from the local PC.

----End

## 5.3.2 Creating a Test Procedure

### Background

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

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.

### Procedure

**Step 1** After the operations described in [5.3.1 Creating a Manual Test Case](#) are complete, choose **Testing Case > Manual Test**, and click the name of the test case to be edited. The **Details** tab page is displayed.

**Step 2** Edit the test details as required.

- In the table under **Test Steps**, click the blank area in the **Test Steps Description** and **Expected Result** columns and enter information as required. Click **+** in the **Operation** column to add a step and enter the step description and expected result as required.
- Click **Associated Requirements** on the right of the page. In the dialog box that is displayed, select the requirement to be associated and click **OK**.

----End

## 5.3.3 Migrating a Manual Test Case

### Background

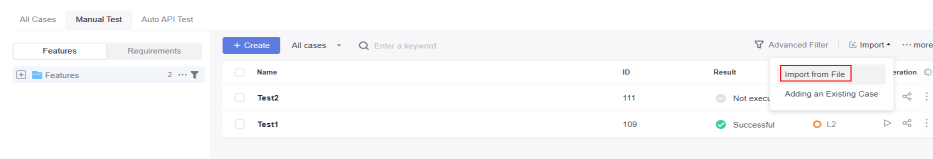
CodeArts TestPlan allows you to import test cases from the local PC to the test case library and export test cases from the test case library.

### Importing a 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**.

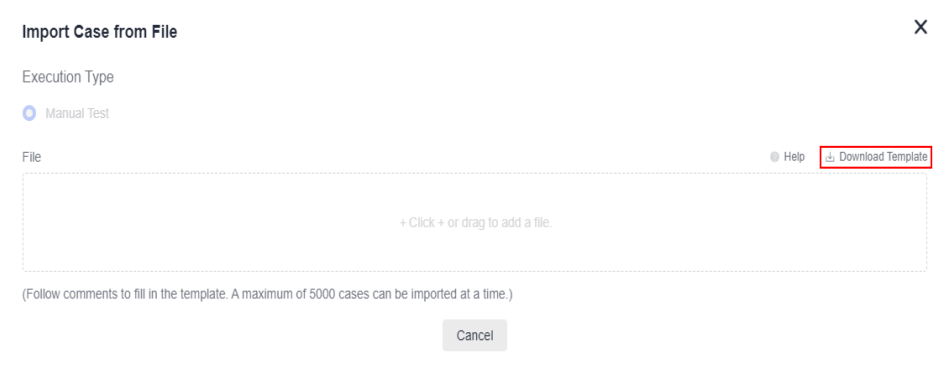
**Step 3** Click the **Manual Test** tab, click **Import** on the right of the page, and choose **Import from File** from the drop-down list.



**Step 4** In the dialog box that is displayed, 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**.



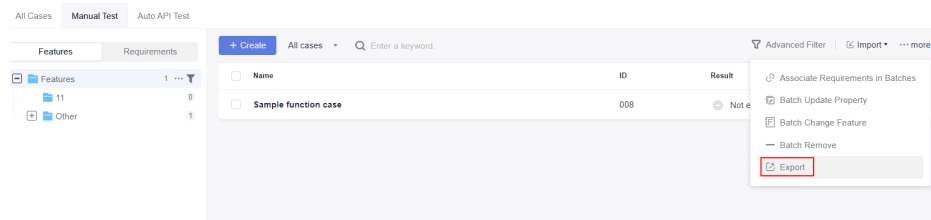
**NOTE**

- A maximum of 5000 test cases can be imported at a time.
- The size of a single file cannot exceed 5 MB.
- Currently, CodeArts TestPlan supports the Excel format. If the data does not meet the import criteria, a message is displayed, asking you to download the error report. Modify the data and import it again.

----End

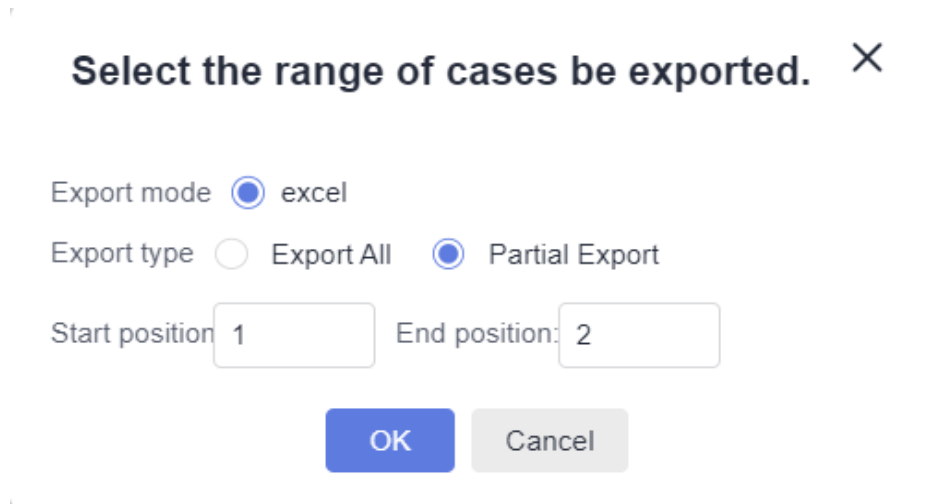
## Exporting a Test Case

**Step 1** On the **Manual Test** tab page, click **More** on the right of the page and choose **Export** from the drop-down list.



**Step 2** In the dialog box that is displayed, select the case export scope. You can select **Export All**.

Select **Partial Export** (set **Start Position** to the case in the first row of the table and **End Position** to the case in the last row of the table by default), and click **OK**.




- Step 3** Open the exported Excel file on the local PC and view the exported test case.  
----End

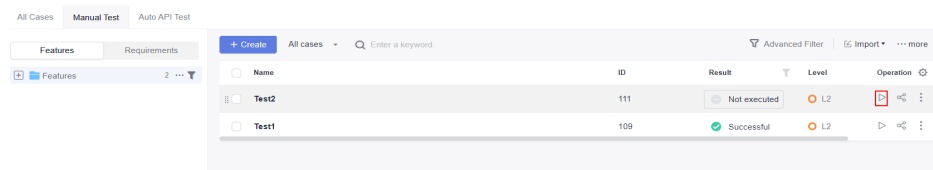
### 5.3.4 Executing a Manual Test Case

#### Prerequisites

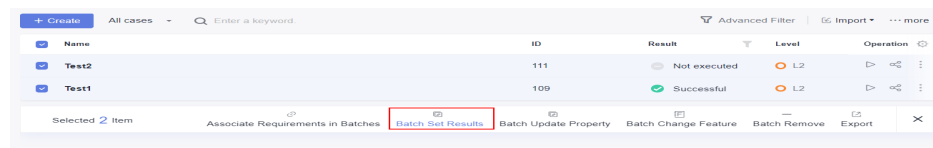
A manual test case has been created.

#### 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** On the **Manual Test** tab page, locate the test case to be executed and click  in the **Operation** column. The execution page is displayed.

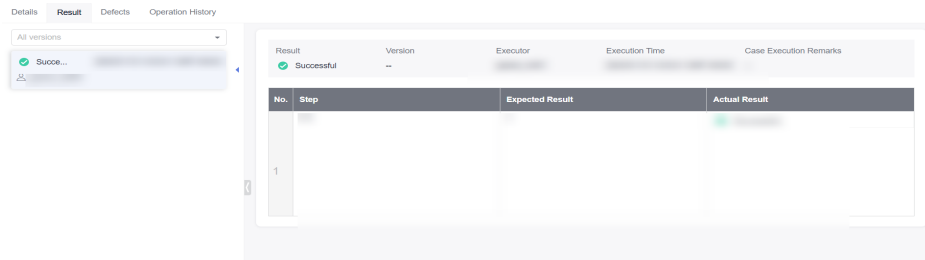


If multiple test cases need to be executed at the same time, select them and click **Batch Set Results**.



- Step 4** Set the case result based on the actual test result.
- Step 5** After the setting is complete, click **Save** to return to the test case list. You can view the execution result in the **Result** column.


**Step 6** Click the test case name and click the **Results** tab to view the execution history of the test case.

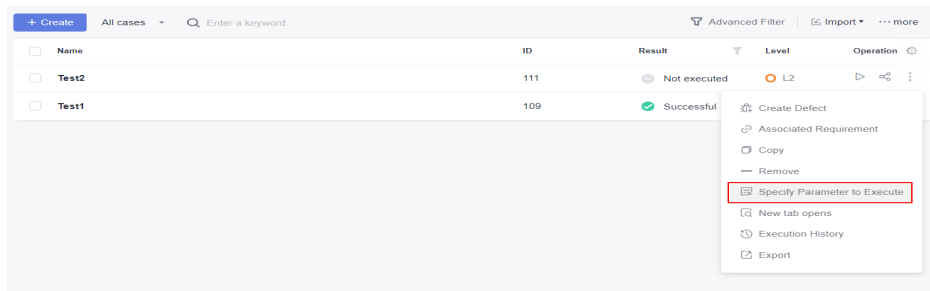


----End

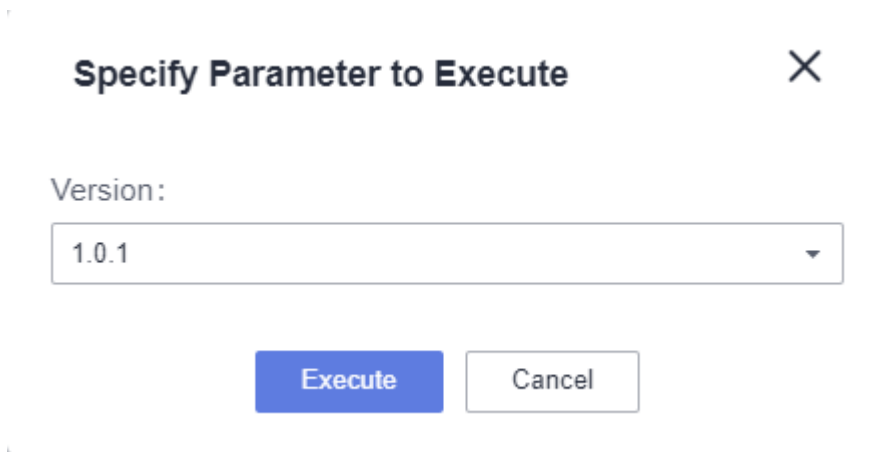
## Execution with Specified Parameters

Manual 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** You can view the test version number. Set the test case execution result, enter the actual result, and click **Save**.

### Execute: test123333

Version: 1.0.1

#### Prerequisites

No prerequisites.

#### Setting Result

\* Expected Result

--Select--  Set the case status to Completed.

**Step 4** Click the corresponding case and click the **Results** tab to query the execution result.

----End

## 5.4 API Automation Test Cases

### 5.4.1 Creating an API Automation Test Case

#### Background

API automation test cases involve the **Details** and **Scripting** tab pages.

- **Details:** manages and describes test cases, including the name (mandatory), type, module, version, iteration, associated requirements, ID, tag, test case level, processor, folder, description, prerequisites, test steps, and expected result.
- **Scripting:** defines the automated testing procedure, including the test steps, logic control, and test parameters.

#### 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** Click the **Auto API Test** tab and click **Create** on the left.

**Step 4** Enter the test case name, configure other information as required, and click **Save**. Alternatively, click **Save and Write Script**. The **Scripting** page is displayed.

The main configuration items of API automation test cases are similar to those of manual test cases. For details, see [5.3.1 Creating a Manual Test Case](#).

 NOTE

The test case name can contain only 3 to 128 characters and allows uppercase letters, lowercase letters, digits, and special characters (-\_/\*&`'^~;:(){}=+,x...—!@#%\$.[]<>?-"').

----End

## 5.4.2 Writing an API Automation Script

### Background

An API automation 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.

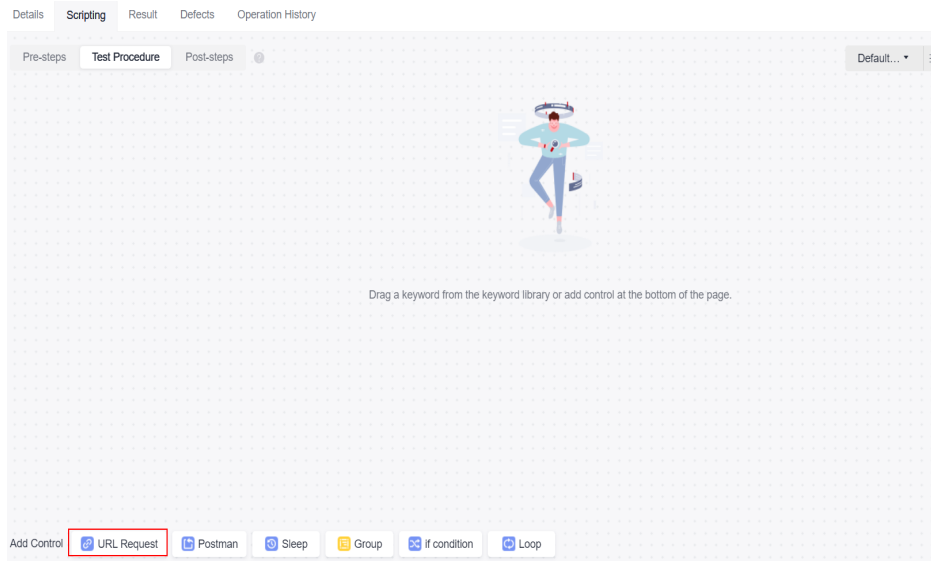
- (Optional) Preparation stage
  - In this stage, prepare the prerequisite data required in the testing stage. If there is no prerequisite, skip this stage.
  - In this stage, the prerequisites are initialized using API calls. If the data of the prerequisites needs to be referenced in the testing stage, you can use parameter passing to parameterize the data. For details, see [5.4.9 Setting Response Extraction](#).
- 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 affecting other tests or the next test, you are advised to clear the test environment data, restore the initial status of the test environment, and destroy the data created in the preparation stage after each test.
  - If no data needs to be destroyed, ignore this stage. You can destroy data using API calls. The test steps in the destruction stage can reference the parameters extracted in the preparation stage.

### Procedure

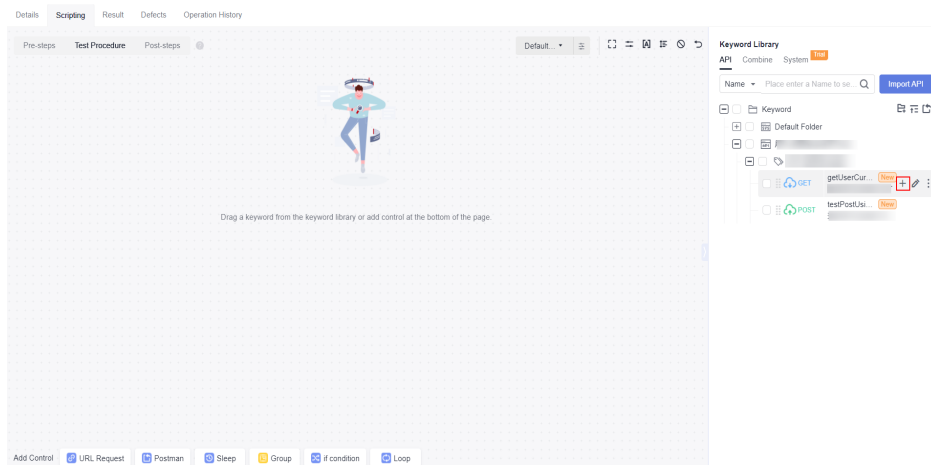
**Step 1** After the operations described in [5.4.1 Creating an API Automation Test Case](#) are complete, choose **Testing Case > Auto API Test** and click the name of the test case to be edited.

**Step 2** Choose **Scripting > URL Request** to generate a test step.



If the Swagger description file of the tested API is available, you can import it to generate a script template, based on which you can orchestrate test cases. For details, see [5.4.4 API Keywords](#).

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 [5.4.7 Setting an API Request](#), [5.4.8 Setting a Test Checkpoint](#), and [5.4.9 Setting Response Extraction](#). Enter a domain name or IP address as the environment address in the request. If you import a Swagger or Postman, the address will be automatically generated to the request.

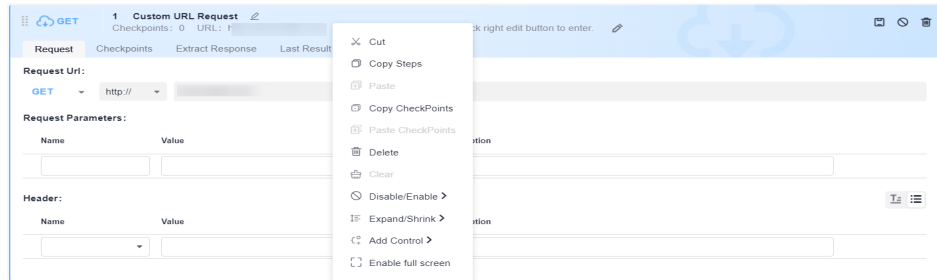
**Step 4** (Optional) Repeat [Step 2](#) to [Step 3](#) to add pre-steps and post-steps.

**Step 5** After the editing is complete, click **Save** in the upper right corner of the page to save the script.

----End

**NOTE**

- API automation 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 [5.6.1 Built-in Functions](#).
- When editing API automation 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 use the "Ctrl+left click" combination 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. Therefore, 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**.

## 5.4.3 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 preparations 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.

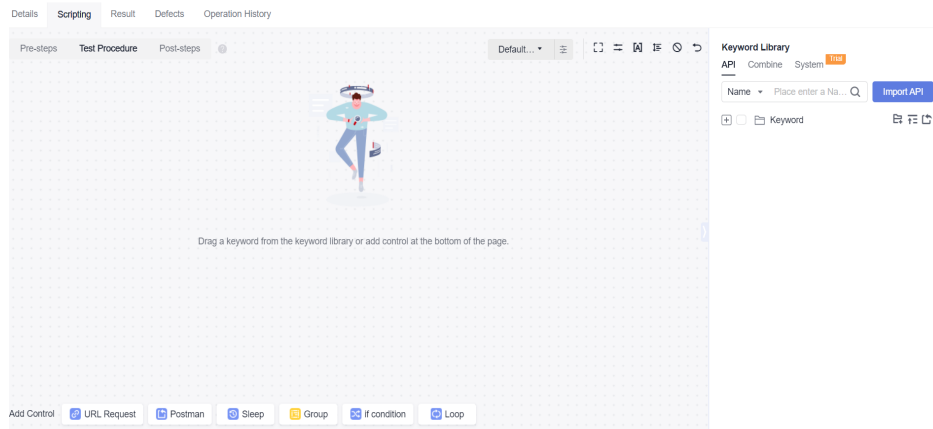
**Keyword Library** manages API keywords and combined keywords in a unified manner to build one-stop keyword management capabilities and ensure consistent user experience during test case scripting.

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. Combined keywords are used to encapsulate common test logic for multiple steps and can be invoked by test cases to implement logic reuse.

### Procedure

- Step 1** After the operations described in [5.4.1 Creating an API Automation Test Case](#) are complete, choose **Testing Case > Auto API Test** and click the name of the test case to be edited.

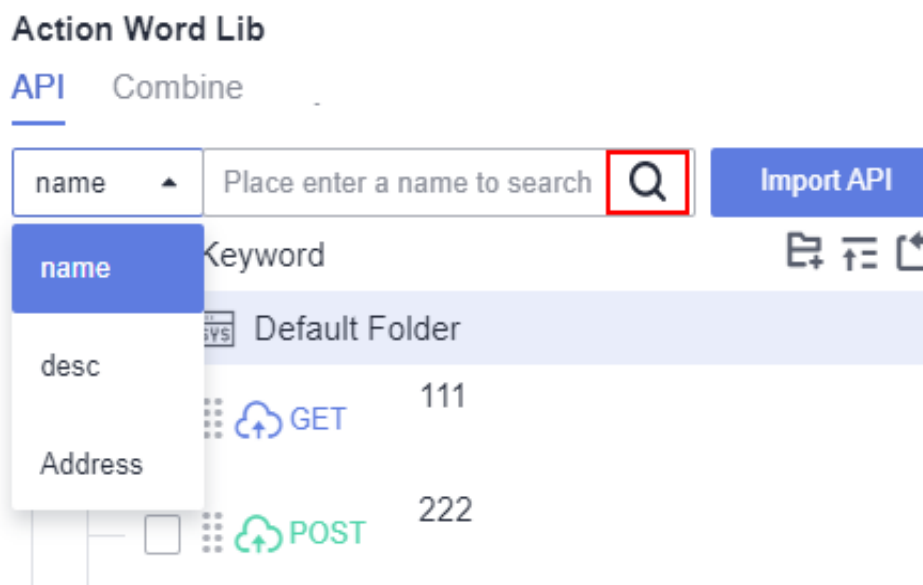
**Step 2** Click the **Scripting** tab. **Keyword Library** is displayed on the right of the page. (If it is not displayed, click )



- The Swagger import keywords are placed on the **API** tab page. For details, see [5.4.4 API Keywords](#).
- Keyword test cases and combined keywords are placed on the **Combine** tab page. You can create combined keywords. For details, see [5.4.5 Combined Keywords](#).

----End

Click the drop-down list on the **API** tab page and search for keywords by **Name**, **Desc**, or **Address**.





## 5.4.4 API Keywords

### 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 API automation 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 mapping between the script template and the fields in the Swagger API description is as follows.

- Swagger 2.0 format is as follows.

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

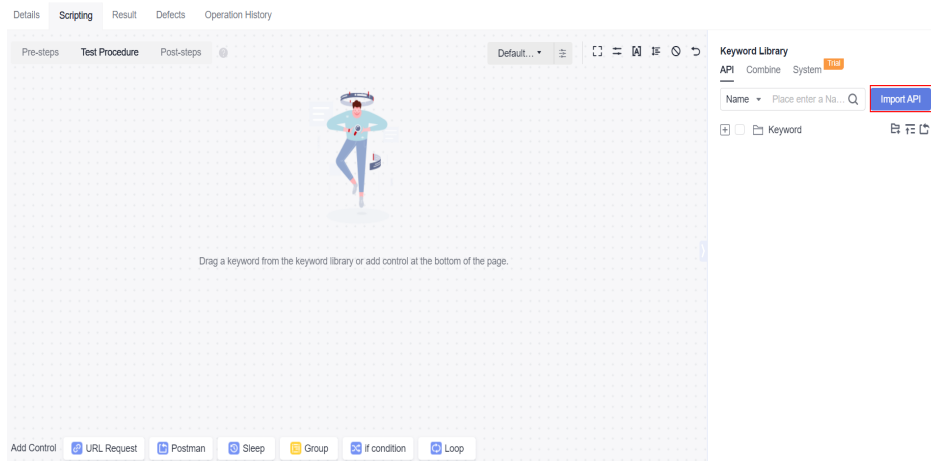
- Swagger 3.0 format is as follows.

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

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

## Importing a Swagger File to Generate a Test Script

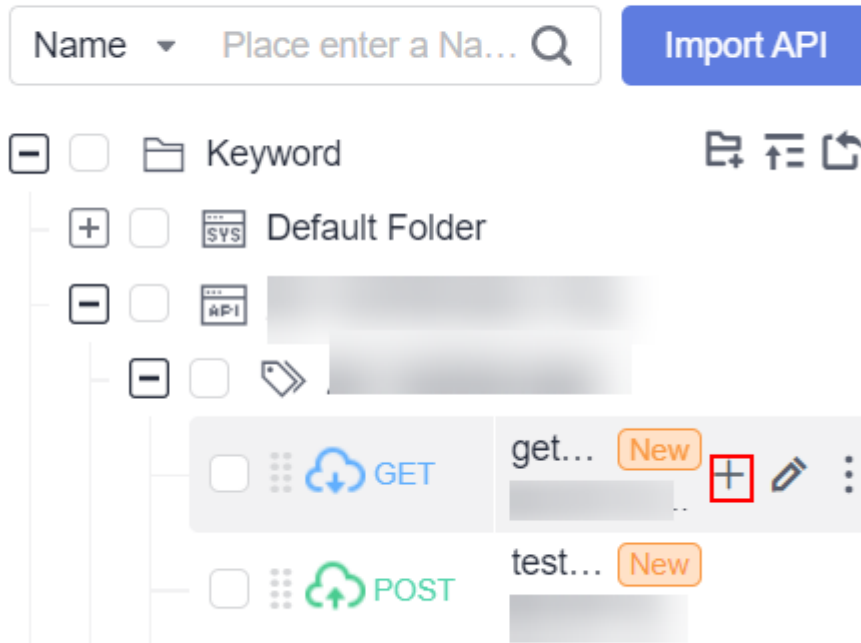
- Step 1** After the operations described in [5.4.1 Creating an API Automation Test Case](#) are complete, click the test case name, and click the **Scripting** tab.
- Step 2** On the **Keyword Library** page on the right, click **Import API**.



- 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.

## Keyword Library


API Combine System Trial

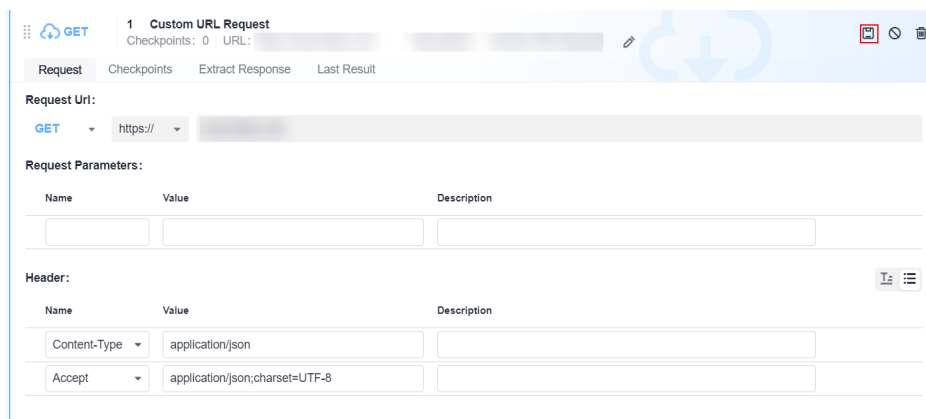


----End

## Saving a Customized API Keyword

**Step 1** After the operations described in [5.4.1 Creating an API Automation Test Case](#) are complete, click the API automation test case name and click the **Scripting** tab.

**Step 2** After the operations described in [5.4.2 Writing an API Automation Script](#) 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 page that is displayed, 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**

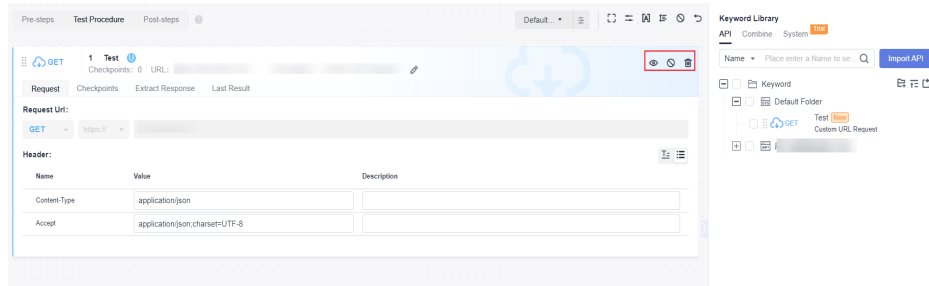
**Save** **Cancel**




**Step 4** Click **Save**.

----End

For an added 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 the test keyword to edit the keyword pair application case details.
- 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.

## 5.4.5 Combined Keywords


### Background

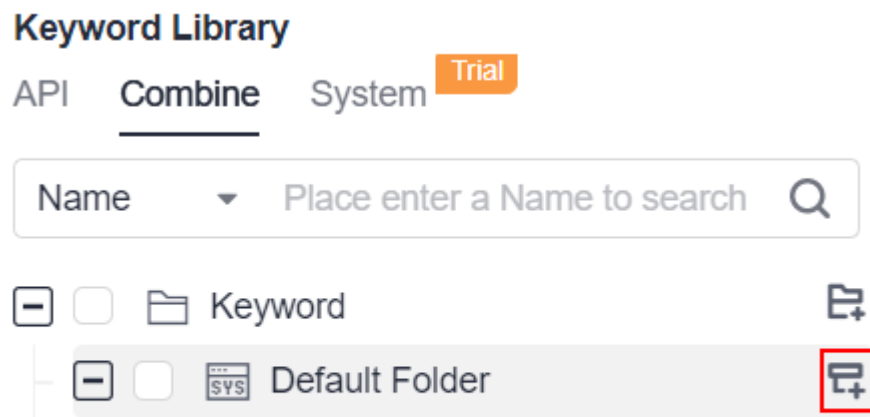
When designing test cases, you may often use the same preparations or test logic. If these steps are written in each test case, the workload is heavy and the maintenance is difficult. Combined keywords are used to encapsulate common test logic for multiple steps and can be invoked by test cases to implement logic reuse.

### 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 the added **URL Request** and click **Save**.
  - Step 3** In the **Keyword Library** > **Combine** tab page, view the combined keywords that have been saved.
- End

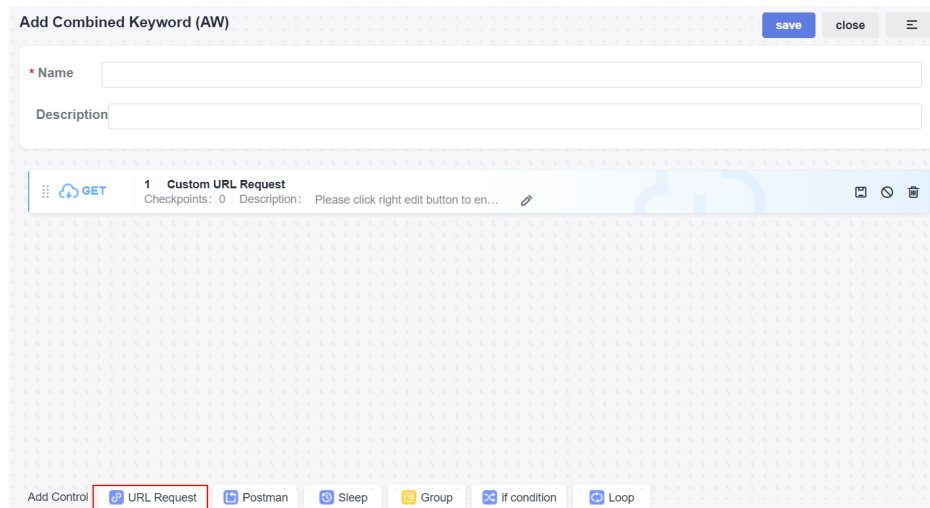
### Scenario 2


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

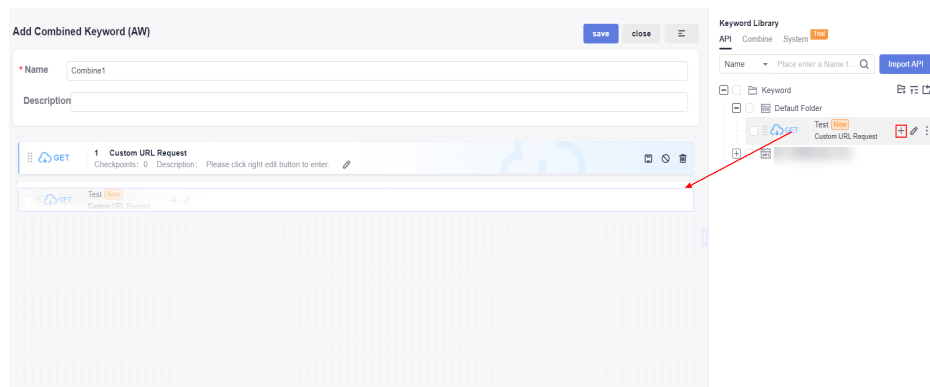


**Step 2** Set **Name** and **Description**.

**Step 3** You can add user-defined URL requests to the combined keywords.





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 hover the cursor over the keyword area and drag the keyword to the test step area.

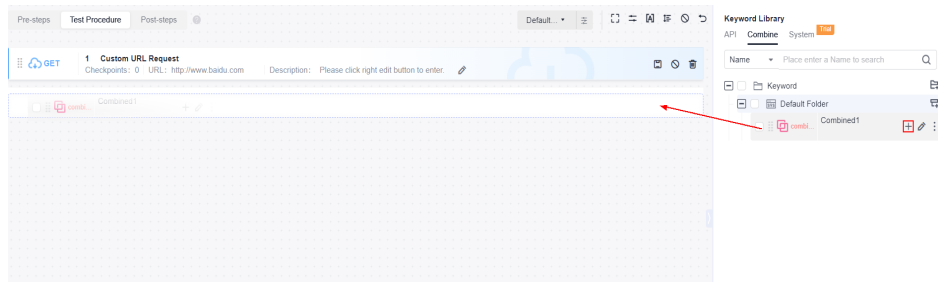



**Step 4 Click Save.**

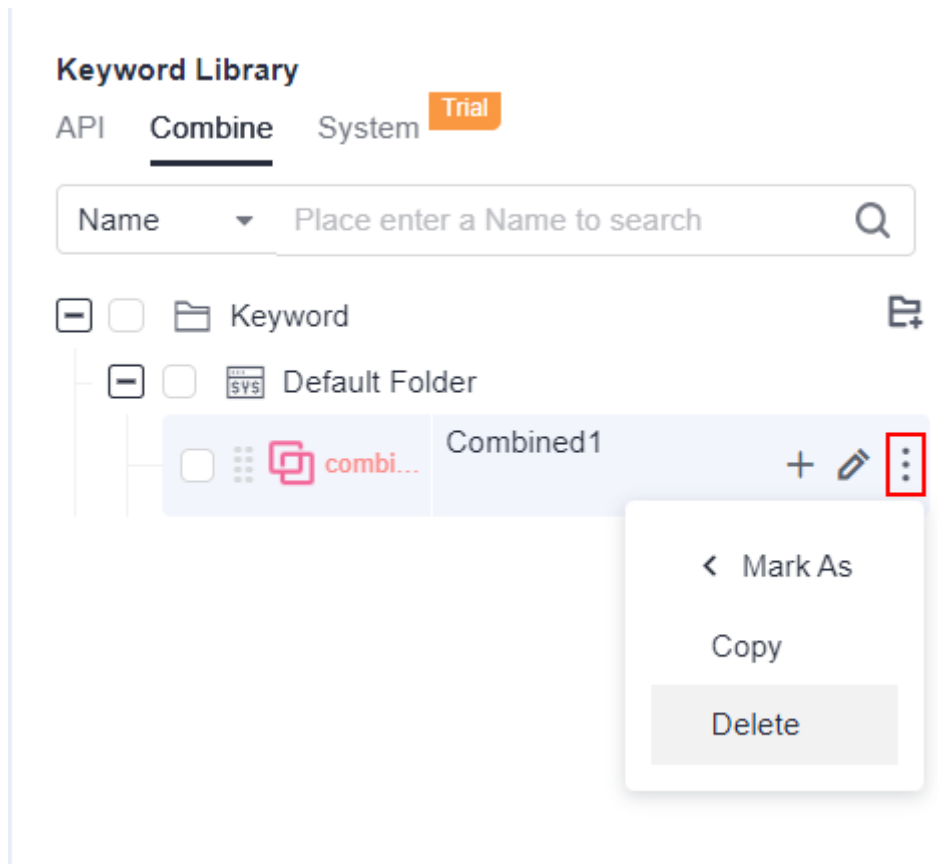
**----End**

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.

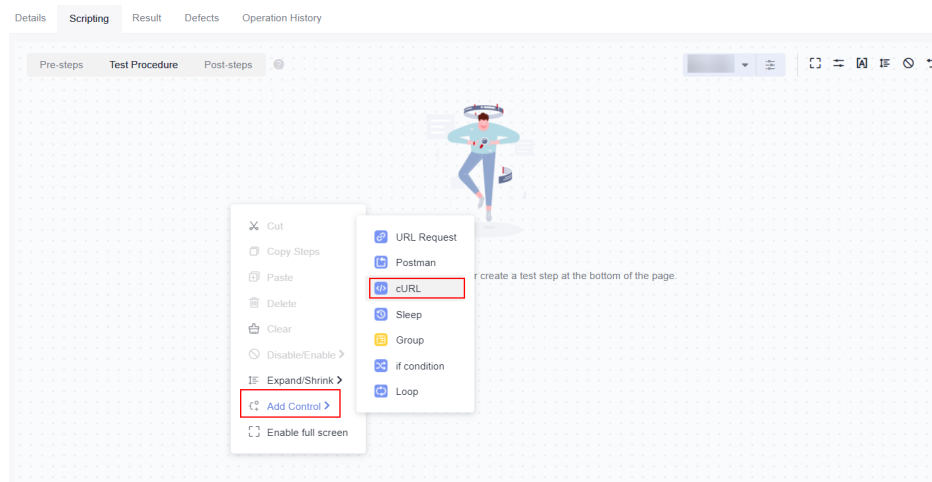
## 5.4.6 Importing cURL to Generate Test Scripts

### 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.

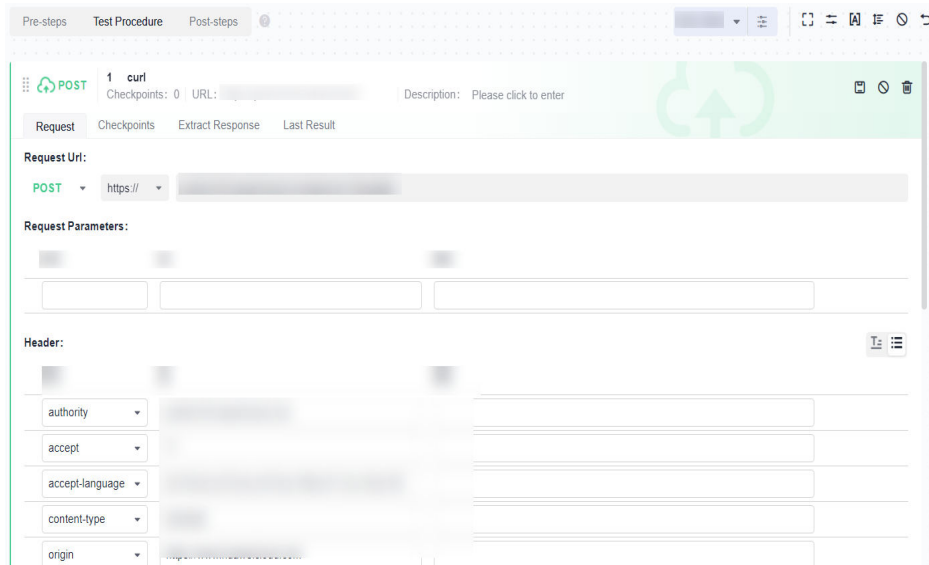
### Procedure

- Step 1** After the operations described in [5.4.1 Creating an API Automation Test Case](#) 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

## 5.4.7 Setting an API Request

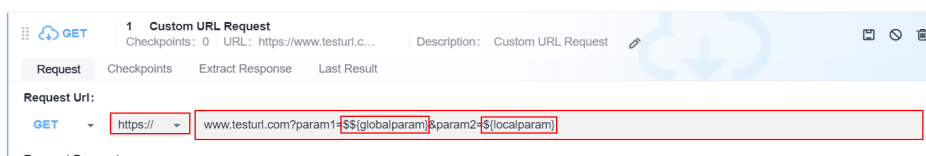
### Requesting URL and URL Parameters

On the **Scripting** tab page of the API automation 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 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 [5.4.12 Setting Test Case Parameters](#).



## Request Header


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

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	

- **Form:** Select or enter a request header name in the **Name** column, and select or enter a value in the **Value** column.

Header: 

Name	Value	Description
Content-Type	application/json	
Accept	application/json;charset=UTF-8	
Accept-Charset		
Accept-Encoding		
Accept-Language		
Accept-Datetime		

- **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
```

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

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.

Request Header	Description
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. Therefore, pay attention to user privacy when deploying this header.
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.

Request Header	Description
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 <b>PUT</b> to update a resource which has not been modified since the last update.
If-Modified-Since	Return <b>304 Not Modified</b> is allowed when the corresponding content is not modified.
If-None-Match	Return <b>304 Not Modified</b> 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 <b>ETag</b> value, which is placed in the <b>ETag</b> field of the HTTP. The client can then decide whether to cache the resource and its <b>ETag</b> . If the client requests the same URL in the future, a request containing the saved <b>ETag</b> and the <b>If-None-Match</b> 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 <b>Access-Control-Allow-Origin</b> 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.
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 <b>Transfer-Encoding</b> in the response protocol header. In addition, the value <b>trailers</b> (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.

Request Header	Description
Via	Request-sending agents informed to the server.
Warning	General warning indicating that errors may exist in the body of an entity.

## 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.

The screenshot shows the 'Request Body' configuration section. At the top, there are four radio buttons: 'none', 'form-data', 'x-www-form-urlencoded', and 'raw'. The 'raw' radio button is selected. To the right of these buttons is a dropdown menu currently set to 'JSON'. A tooltip is visible over the dropdown menu, showing 'Text' and 'JSON' as options. Below the radio buttons, there is a text input field with the number '1' and a small icon to its left.

- Form: The text and file types are supported.
  - Text: Set the parameter name and value.

The screenshot shows the 'Request Body' configuration section with the 'form-data' radio button selected. Below the radio buttons is a table with three columns: 'Type', 'Name', and 'Value'. The 'Type' column has a dropdown menu set to 'Text'. The 'Name' column has a text input field containing 'date'. The 'Value' column has a text input field containing '2023-3-30'.

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

The screenshot shows the 'Request Body' configuration section with the 'form-data' radio button selected. Below the radio buttons is a table with three columns: 'Type', 'Name', and 'Value'. The 'Type' column has a dropdown menu set to 'File'. The 'Name' column has an empty text input field. The 'Value' column has a dropdown menu set to '--select--' and a red box around a vertical ellipsis icon (three dots) to its right.

## 5.4.8 Setting a Test Checkpoint

### Suggestions

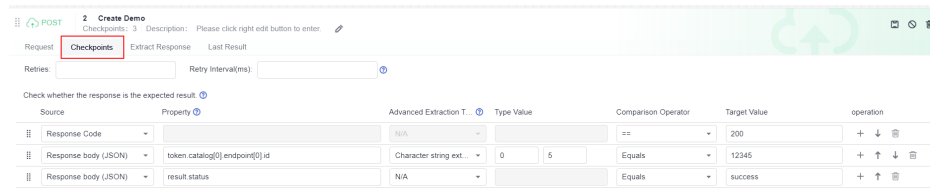
You are advised to set checkpoints. For API requests, you need to provide checkpoints for determining response codes.

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.
Retry Interval	Interval for retrying a checkpoint if the checkpoint fails. The unit is ms.
Source	Source of the detected field, such as the response body (JSON), response header, response code, and variable.

Parameter	Description
Property	<ul style="list-style-type: none"><li>• If the source is a response code, this parameter can be left empty. For details, see <a href="#">Response Code Check</a>.</li><li>• If the source is a response header, the property is the name of the field in the response header. For details, see <a href="#">Response Header Check</a>.</li><li>• If the source is the response body (JSON), the property can be set in either of the following ways:<ol style="list-style-type: none"><li>1. Common extraction expression (not starting with \$), for example, <b>item.name</b>. Obtain the value of a field. Nested values are supported. For details, see <a href="#">Response Body (JSON) Check</a>. When an array is extracted from the response body, the index can be a number or a <b>key:value</b> expression. For details, see <a href="#">Example: Obtaining a String from the Response Body Based on the Specified key:value</a>.</li><li>2. JSONPath expression (starting with \$. or \$[]), for example, <b>\$.store.book[0].title</b>. For details, see <a href="#">Obtaining Data from the Response Body Based on JSONPath</a>.</li></ol></li><li>• If the source is a variable, the property is a global variable, a local variable, or a variable after response extraction. For details, see <a href="#">Variables Check</a>.</li></ul>
Advanced Extraction Type	<p>(Optional) Use the advanced extraction type to assist in extracting checkpoint information. If <b>N/A</b> is selected, no additional matching mode is used.</p> <p>Currently, there are two modes:</p> <ul style="list-style-type: none"><li>• <a href="#">5.6.2.1 Character String Extraction</a>: truncates strings.</li><li>• <a href="#">5.6.2.2 Regular Expression</a>: filters source strings using regular expressions.</li></ul> <p>For the advanced extraction type, the string extraction function is preferred. If the function cannot meet the requirements, you can use the regular expression.</p>
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 <a href="#">Comparison Operator Description</a> .
Target Value	Expected value of a checkpoint. Built-in parameters can be used for target values. For details, see <a href="#">Built-in Parameters</a> .

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

## Comparison Operator Description

CodeArts TestPlan supports the following comparison types.

Comparison Type	Comparison Operator	Value Mandatory or Not	Example
Digit	<ul style="list-style-type: none"><li>• ==</li><li>• !=</li><li>• &gt;=</li><li>• &lt;=</li><li>• &gt;</li><li>• &lt;</li></ul>	Yes	<ul style="list-style-type: none"><li>• Response code = 200</li><li>• Response code != 200</li><li>• Response code &gt;= 200</li><li>• Response code &lt;= 200</li><li>• Response code &gt; 200</li><li>• Response code &lt; 200</li></ul>

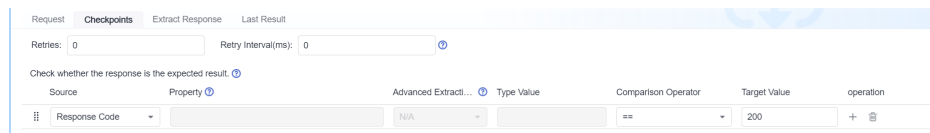


Comparison Type	Comparison Operator	Value Mandatory or Not	Example
String	<ul style="list-style-type: none"> <li>• Equals</li> <li>• Does not equal</li> <li>• Equals (case insensitive)</li> <li>• Contains</li> <li>• Does not contain</li> </ul>	Yes	<ul style="list-style-type: none"> <li>• Parameter 1 in the response body equals <b>test</b>.</li> <li>• Parameter 2 in the response body does not equal <b>test</b>.</li> <li>• Parameter 3 in the response body equals <b>TEST</b>.</li> <li>• Parameter 4 in the response body contains <b>tri</b>.</li> <li>• Parameter 5 in the response body does not contain <b>tri</b>.</li> </ul>
Regular expression	Regular expression	Yes	<ul style="list-style-type: none"> <li>• Regular expression for parameter 1 in the response body: <b>^[A-Za-z0-9]{1,32}\$</b></li> </ul>
General	<ul style="list-style-type: none"> <li>• Is empty</li> <li>• Is not empty</li> </ul>	No	<ul style="list-style-type: none"> <li>• Parameter 1 in the response body is empty.</li> <li>• Parameter 2 in the response body is not empty.</li> </ul>
JSON array	<ul style="list-style-type: none"> <li>• Has null JSON array</li> <li>• Has non-null JSON array</li> </ul>	No	<ul style="list-style-type: none"> <li>• Parameter 1 in the response body has a null JSON array.</li> <li>• Parameter 2 in the response body has a non-null JSON array.</li> </ul>
	JSON array size	Yes	<ul style="list-style-type: none"> <li>• Size of the parameter 1 <b>JSON</b> array in the response body.</li> </ul>
Type	<ul style="list-style-type: none"> <li>• Is JSON type</li> <li>• Is JSON array type</li> </ul>	No	<ul style="list-style-type: none"> <li>• Parameter 1 in the response body is of the JSON type.</li> <li>• Parameter 2 in the response body is of the JSON array type.</li> </ul>

Comparison Type	Comparison Operator	Value Mandatory or Not	Example
JSON object	JSON value	Yes	<ul style="list-style-type: none"> <li>The JSON value of parameter 1 in the response body is <code>{"name":"zhangsan"}</code>.</li> </ul>

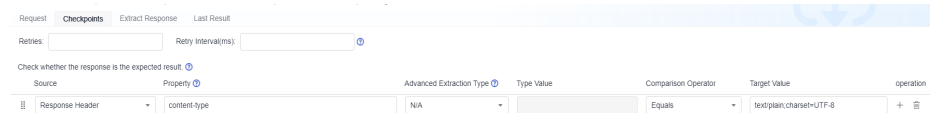
## Response Code Check

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



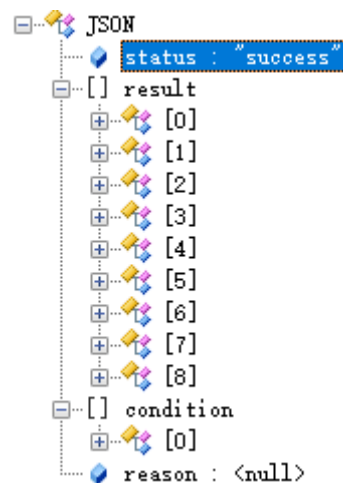
## Response Header Check

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**.

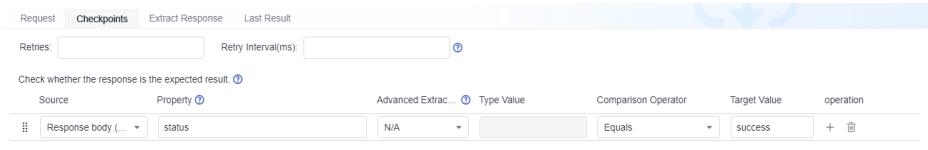


## Response Body (JSON) Check

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

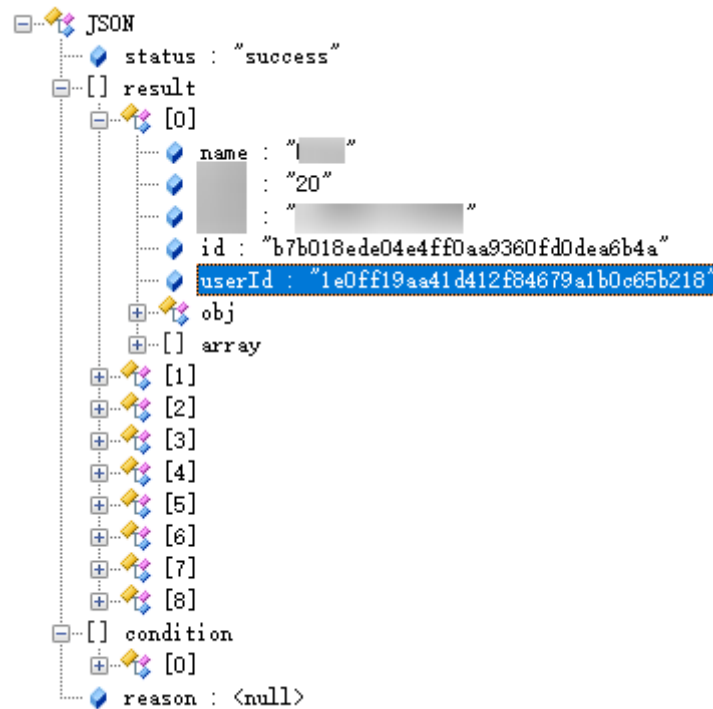


Check the value of the field whose name is **status** in the response body object. The checkpoint configuration is as follows.

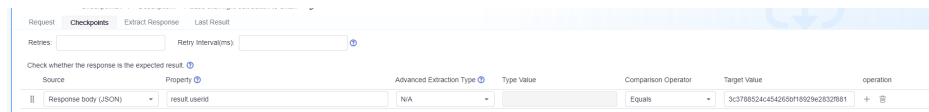


2. Check the field value of an array object in the response body (JSON). (The array condition uses the subscript starting from 0 to determine the object.) For example:

The response body structure is as follows.



Check the value of **userId** in the first element object field of the **result** array in the response body. The checkpoint configuration is as follows.



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.



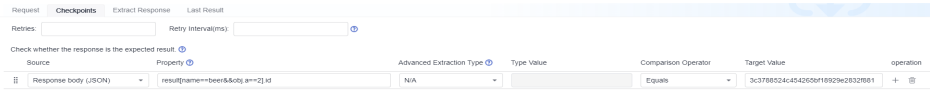
- a. Check all objects whose **name** is **beer** in the **result** array of the response body. Obtain the values of **id** after the first object. The checkpoint configuration is as follows.

Source	Property	Advanced Extraction Type	Type Value	Comparison Operator	Target Value	operation
Response body (JSON)	result[name=beer][0].id	N/A		Equals	3c3788524c4542558f15929e283285f	

#### 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**.

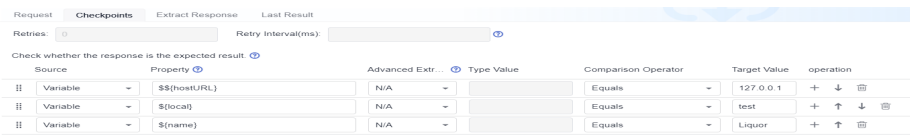
- b. Check the ID of the object whose **name** in the **result** array of the response body is **beer**, and of which the **a** property under **obj** is **2**. The checkpoint configuration is as follows.



## 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**.



## 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
<=	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==**.

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 <b>name</b> is <b>Liquor</b> from the <b>result</b> array.	result[name==Liquor]
Obtain the data whose <b>name</b> is not <b>Beer</b> and <b>quantity</b> is greater than 20 from the <b>result</b> array.	result[name!=Beer&&quantity>20]
Obtain the data whose <b>name</b> is a null object and <b>id</b> is less than or equal to 2 from the <b>result</b> array.	result[name==\$null&&array[id<=2]]
Obtain the data whose <b>name</b> is a null string and property <b>a</b> under <b>obj</b> is 2 from the <b>result</b> array.	result[name==null&&obj.a==2]
Obtain the data whose <b>name</b> is an empty string ("" ) from the <b>result</b> 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
\$.store.book[*].author	The authors of all books.
\$.author	All authors.
\$.store.*	All things, both books and bicycles.
\$.store..price	The price of everything.
\$.book[2]	The third book.
\$.book[-2]	The second to last book.

JSONPath Expression	Description
<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 number.
<code>\$.store.book[?(@.price &lt; 10)]</code>	All books in store cheaper than 10.
<code>\$.book[?(@.price &lt;= \$('[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

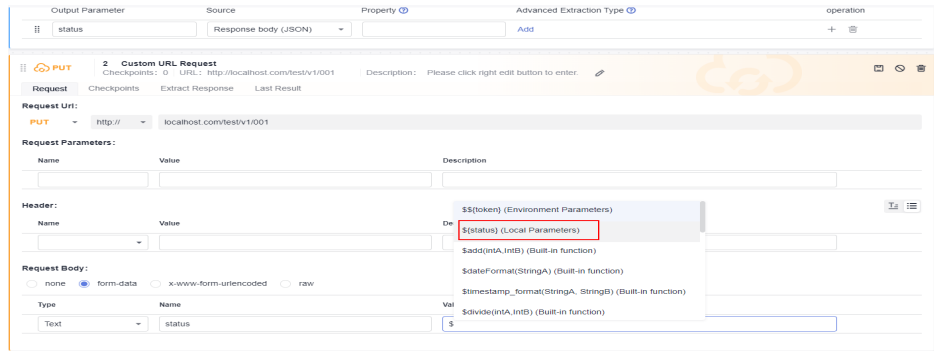
If the **length** or **size** function is used, do not use deep scanning (that is, the `..` symbol) when calling the function for multiple times. For example, if `$.book.length()` returns the number of books, you need to determine the path and change it to `$.store.book.length()`.

## 5.4.9 Setting Response Extraction

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.

- 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	<b>`\${Output Parameter}</b> , 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.
Property	<ul style="list-style-type: none"> <li>• If the source is a response code, this parameter can be left empty. For details, see <a href="#">Response Code Check</a>.</li> <li>• If the source is a response header, the property is the name of the field in the response header. For details, see <a href="#">Response Header Check</a>.</li> <li>• If the source is the response body (JSON), the property can be set in either of the following ways: <ol style="list-style-type: none"> <li>1. Common extraction expression (not starting with \$), for example, <b>item.name</b>. Obtain the value of a field. Nested values are supported. For details, see <a href="#">Response Body (JSON) Check</a>. When an array is extracted from the response body, the index can be a number or a <b>key:value</b> expression. For details, see <a href="#">Example: Obtaining a String from the Response Body Based on the Specified key:value</a>.</li> <li>2. JSONPath expression (starting with \$. or \$[]), for example, <b>\$.store.book[0].title</b>. For details, see <a href="#">Example: Obtaining Data from the Response Body Based on JSONPath</a>.</li> </ol> </li> </ul>

Parameter	Description
Advanced Extraction Type	(Optional) Use the advanced extraction type to assist in extracting response information. If <b>N/A</b> is selected, no additional matching mode is used. Currently, there are two modes: <ul style="list-style-type: none"><li>• <b>5.6.2.1 Character String Extraction</b>: truncates strings.</li><li>• <b>5.6.2.2 Regular Expression</b>: filters source strings using regular expressions.</li></ul> 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 <b>dynamic parameter</b> can be referenced in subsequent tests.

## 5.4.10 Importing a Postman File

### Background


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

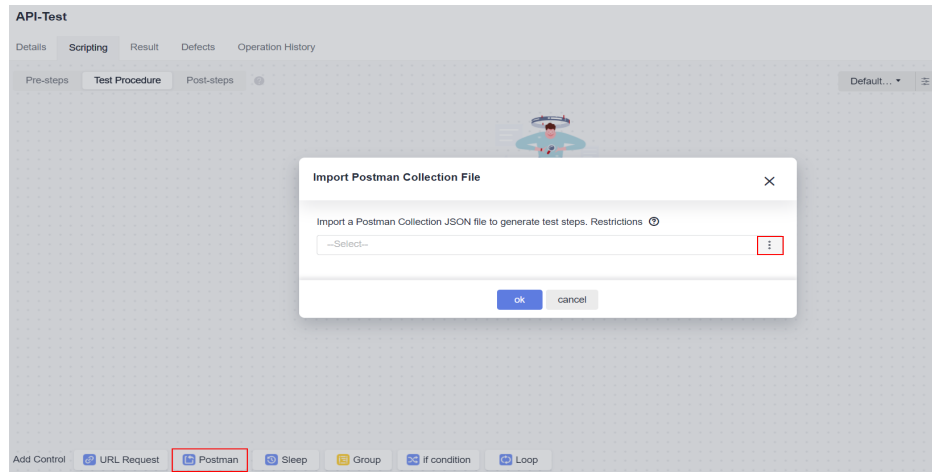
Requirements:

- 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.

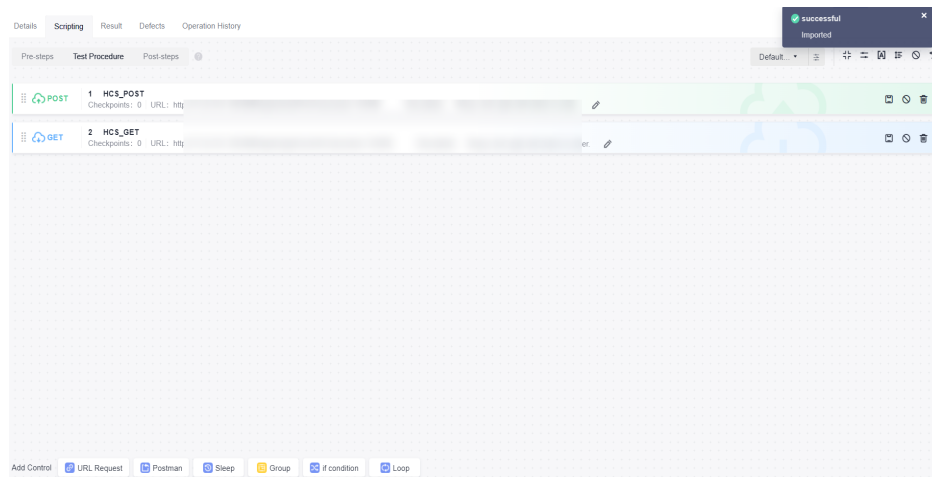
### Procedure

**Step 1** After the operations described in [5.4.1 Creating an API Automation Test Case](#) are complete, click the test case name, and click the **Scripting** tab.

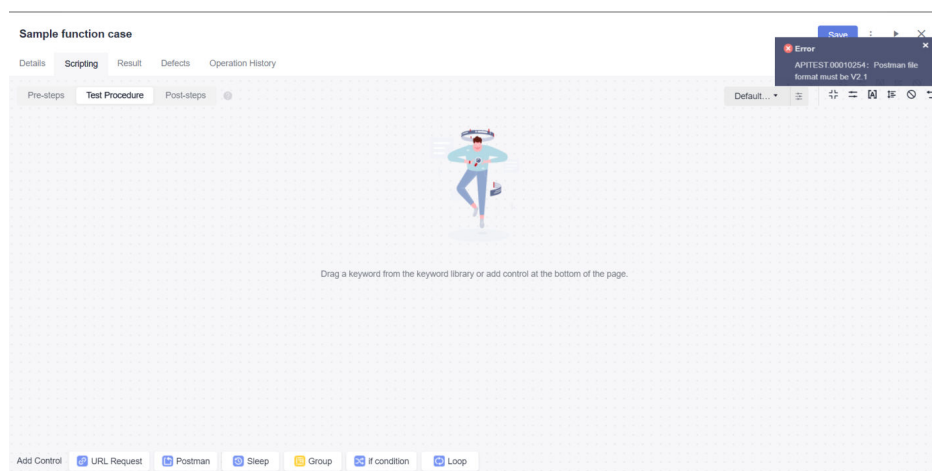
**Step 2** Click **Postman**. In the dialog box that is displayed, click , select the file to be imported, and click **ok**.



**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

## 5.4.11 Inserting Logic Control

### Background

Logic control is used to orchestrate test scenarios, including:

- [Sleep](#)
- [Group](#)
- [If Condition](#)
- [Loop](#)

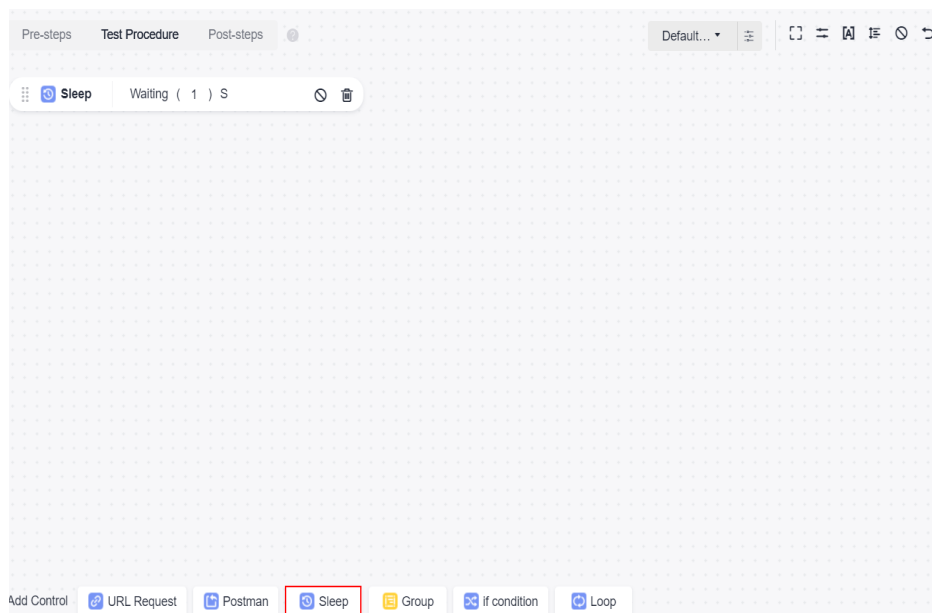
### Sleep

**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.



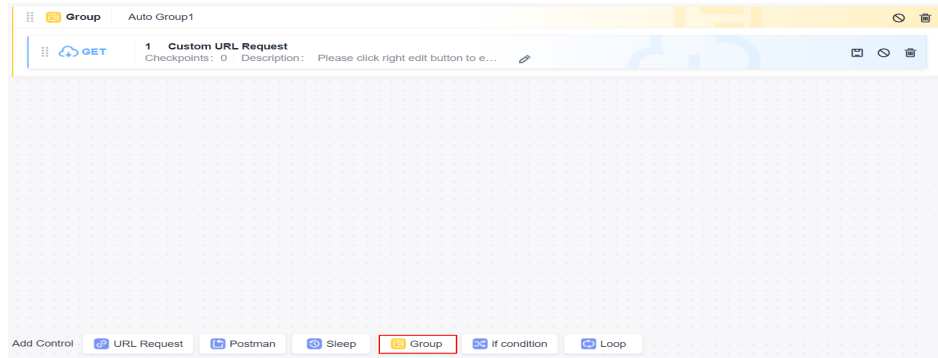
----End

### 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

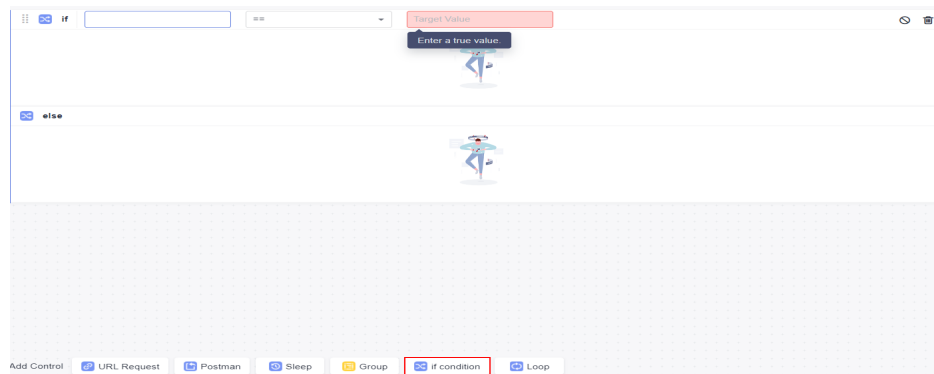
If you need to determine the subsequent test steps based on the results of the previous test steps, you can use **If Condition**.

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

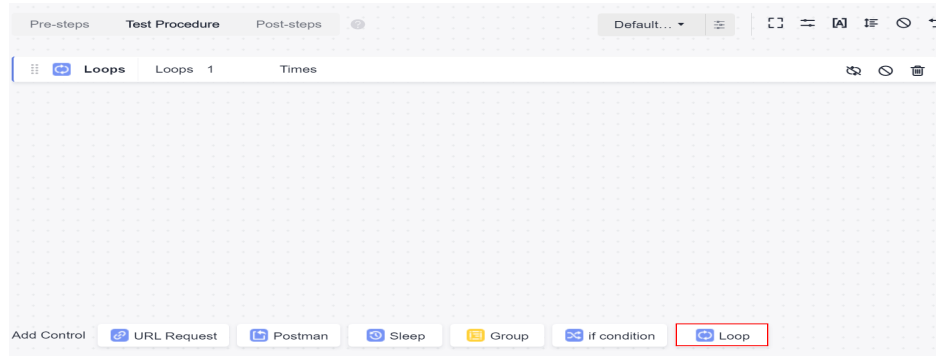
The **if** condition supports referencing local variables and passing parameters. For details about how to use comparison operators, see [Comparison Operator Description](#).

## Loop

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

**Step 2** Set the loop count.

**URL Request, If Condition, Sleep, and Loop** can be added to a loop.



----End

## 5.4.12 Setting Test Case Parameters

### 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 API automation test cases are classified into the following types:

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

### 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 main configuration items of local parameters are as follows.

**Table 5-1** Main local parameter configuration items

Configur ation Item	Ma nda tor y	Description
Name	Yes	The value can contain 1 to 300 characters. Only letters, digits, periods (.), underscores (_), and hyphens (-) are supported.



Configuration Item	Mandatory	Description
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 <a href="#">Table 5-2</a> .
Description	No	Brief description of the parameter (current parameter by default). The value can contain up to 3000 characters. Click the text box to enter the description.  Click  , enter a JSON string, click <b>JSON Parse</b> to add line breaks and indentation, and click <b>Fill Back</b> .
Value	No	Local parameter values. Assign values to the supported parameter types by referring to <a href="#">Table 5-2</a> .
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 <b>Extract Response</b> tab page of a test step, set the value of the <b>Assign Value to Dynamic Environment Parameter</b> column. The extracted value will be assigned to the dynamic parameter during test execution.

**Table 5-2** Parameter types

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

Name	Description
Random integer	The system randomly generates an integer from -999999999 to 999999999. The parameter of this type cannot be set to <b>Sensitive</b> or <b>Dynamic</b> . 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 <b>Sensitive</b> or <b>Dynamic</b> .
Formatted timestamp	The format is yyyy-MM-dd HH:mm:ss or yyyy-MM-dd. The parameter of this type cannot be set to <b>Sensitive</b> or <b>Dynamic</b> . 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 <b>Sensitive</b> or <b>Dynamic</b> .
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 <b>Sensitive</b> or <b>Dynamic</b> .
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 <b>Sensitive</b> or <b>Dynamic</b> .
Password or authentication information	The value can contain up to 256 characters. The parameter of this type is set to <b>Sensitive</b> 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 <b>Sensitive</b> by default and cannot be changed.

To set local parameters, perform the following steps:

**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

## 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	<ul style="list-style-type: none"><li>Checkpoint property field.</li><li>Property field for parameter passing.</li></ul>	<ul style="list-style-type: none"><li>Checkpoint: Check whether the value of <b>id</b> in the response body is <b>100</b>.</li><li>Settings: Set <b>Source</b> to <b>Response body (JSON)</b>, <b>Property</b> to <b>id</b> (only if the <b>id</b> field exists in the JSON string), <b>Comparison Operator</b> to <b>Equals (case insensitive)</b>, and <b>Target Value</b> to <b>100</b>.</li></ul>
Response Header	Response header returned by an API.	Yes	<ul style="list-style-type: none"><li>Checkpoint property field.</li><li>Property field for parameter passing.</li></ul>	<ul style="list-style-type: none"><li>Checkpoint: Check whether the value of <b>token</b> in the response header is <b>abcd</b>.</li><li>Settings: Set <b>Source</b> to <b>Response Header</b>, <b>Property</b> to <b>token</b> (only if <b>token</b> exists in the response header), <b>Comparison Operator</b> to <b>Equals (case insensitive)</b>, and <b>Target Value</b> to <b>abcd</b>.</li></ul>
Response Code	Response code returned by an API.	No	<ul style="list-style-type: none"><li>Property or value of a checkpoint.</li><li>Property field of a variable.</li></ul>	<ul style="list-style-type: none"><li>Checklist: Check whether the response code is <b>200</b>.</li><li>Settings: Set <b>Source</b> to <b>Response Code</b>, <b>Comparison Operator</b> to <b>Equals</b>, and <b>Target Value</b> to <b>200</b>.</li></ul>

**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 [5.4.9 Setting Response Extraction](#).

## 5.4.13 Setting Environment Parameters

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 parameterization 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.

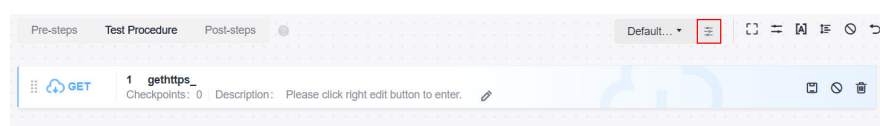
## Reference Format

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 API automation test cases can be reused in all environments.

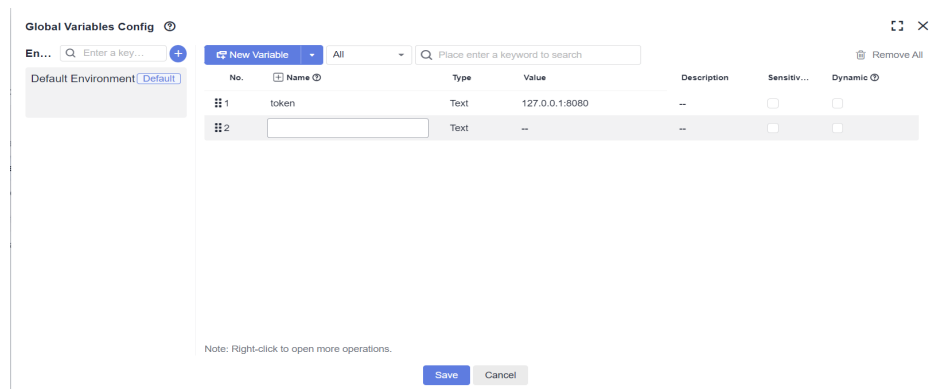
## Procedure

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




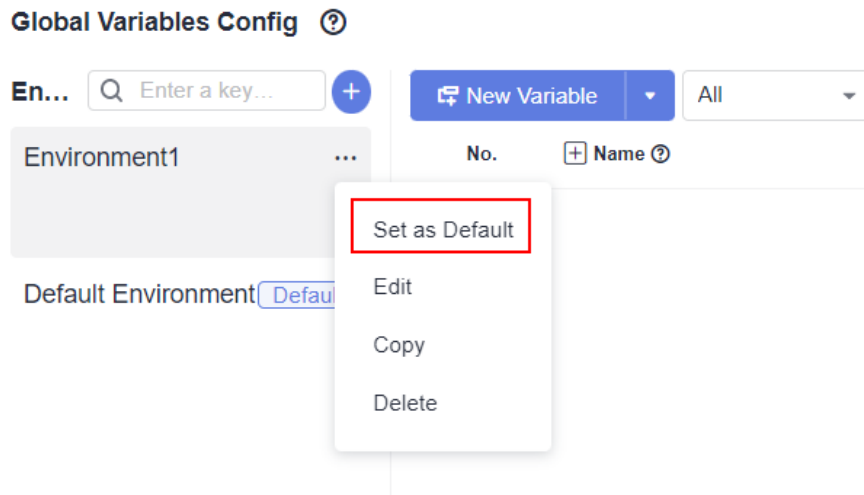
- Step 2** Click **New Variable**, enter parameter information, and click **OK**.

Configuration Item	Description
Name	Parameter name.
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 <b>Assign Value to Dynamic Environment Parameter</b> column in the <b>Extract Response</b> tab page of the test step, the extracted value is assigned to the dynamic parameter during test execution. For details, see <a href="#">Dynamic Variable Description</a> .



**Step 3** Change the default environment.

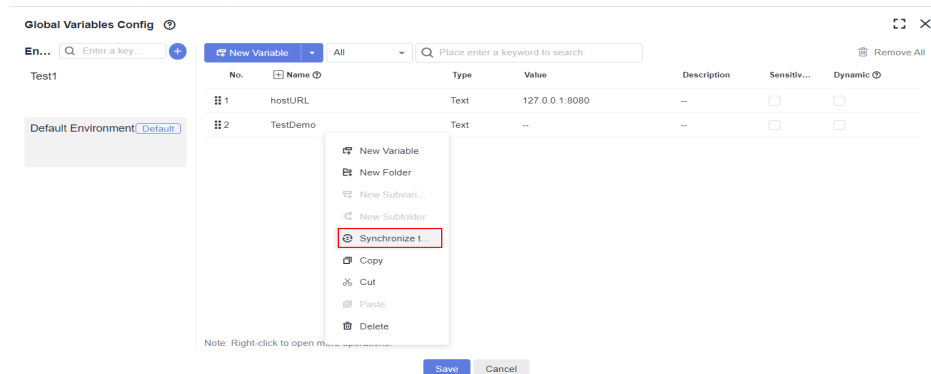
To set another environment as the default environment, click  in the upper right corner of the environment card and select **Set as Default**.



----End

## 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.



## Dynamic Variable Description

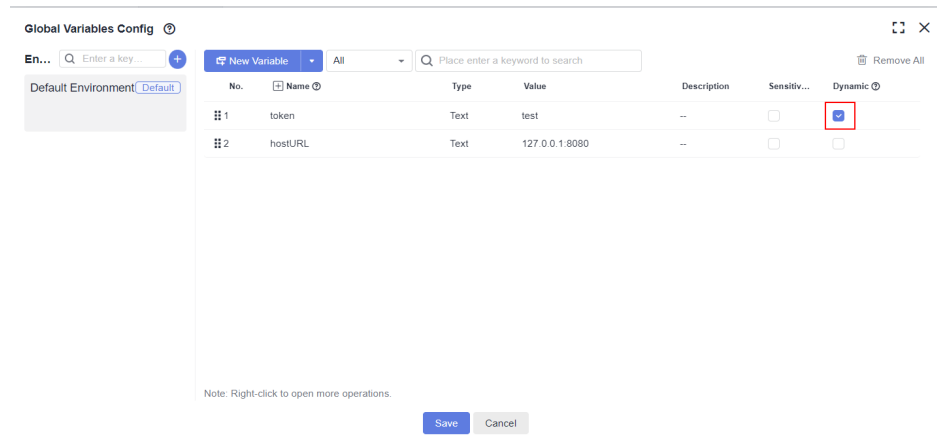
Multiple test cases in a test suite may have context relationships. The test case that is executed later depends on the result returned by the API in the test case that is executed earlier. For example, all APIs require authentication information, which has a validity period. If it is obtained in each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

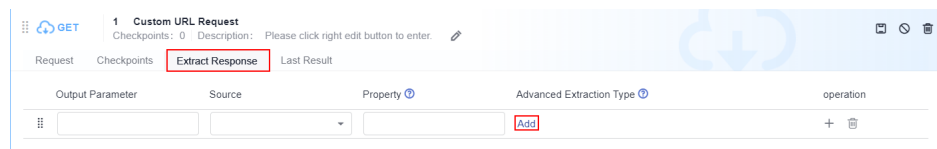
1. On the test case list of the **Auto API Test** tab page, click **More** on the right of the page and choose **Environment Parameters**.

2. Select the check box in the **Dynamic** column and click **Save** to set a global variable to a dynamic variable.

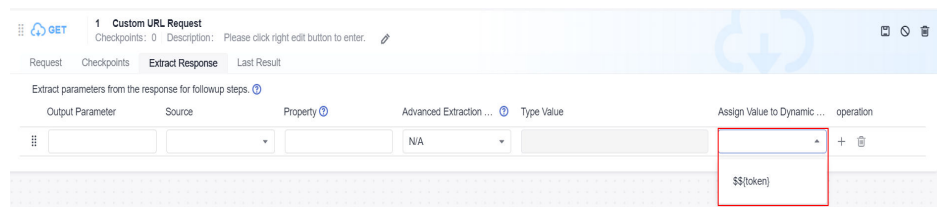


### Step 2 Assign values to dynamic variables.

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



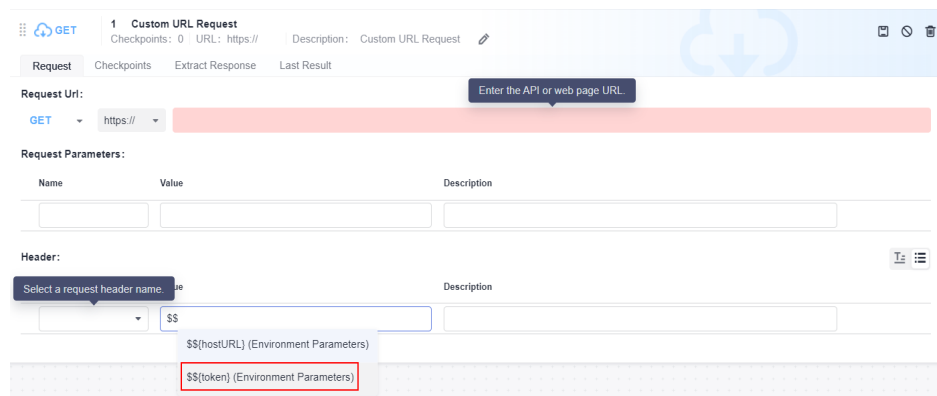
2. Select a value from the **Assign Value to Dynamic Environment Parameter** drop-down list box.



### 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

## 5.4.14 Importing an API Automation Case

### Background

CodeArts TestPlan allows you to generate test cases by importing files in any of the following formats:

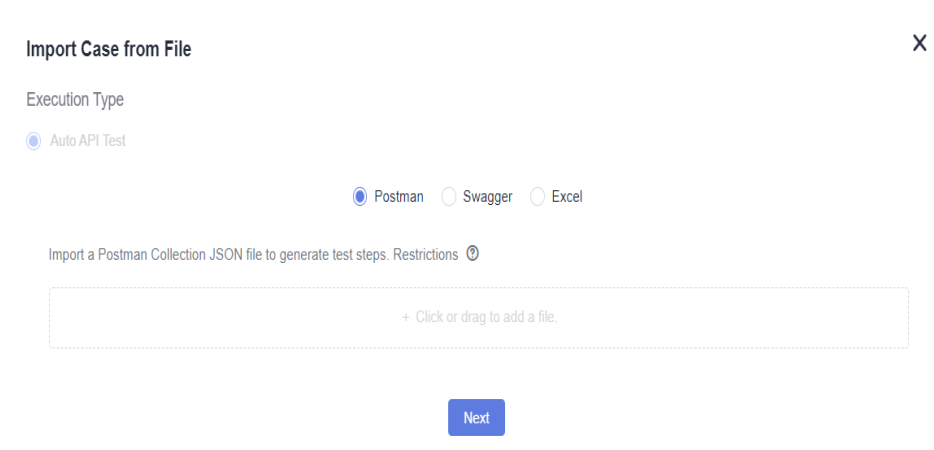
- Postman: Postman Collection v2.1 standard, Postman Collection JSON file
- Swagger: Swagger 2.0 and 3.0 standards, YAML file
- Excel: Excel file based on the given template

### Importing a Postman or Swagger File

Only one test case can be imported at a time. In addition, only test steps can be generated for the imported test case. Preparations and followups are not supported.

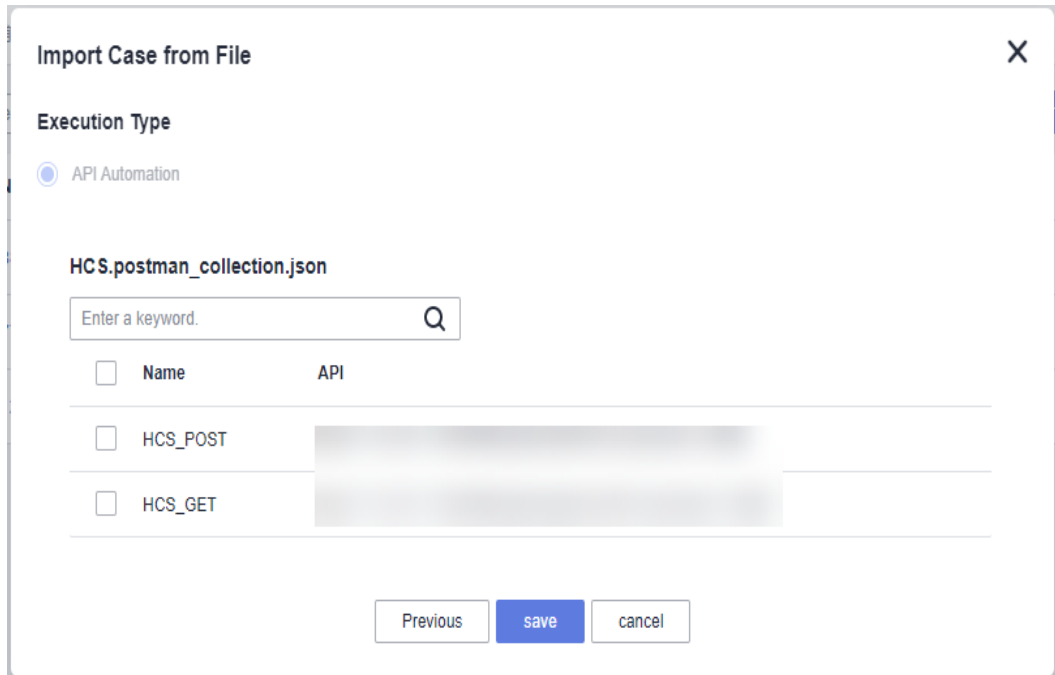
- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** Click the **Auto API Test** tab and choose **Import > Import from File** on the right of the page. The **Import Case from File** window is displayed.
- Step 4** 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 5** In the displayed list, select the items for which you want to generate test cases based on the step sequence and click **save**.



----End

## Importing an Excel File

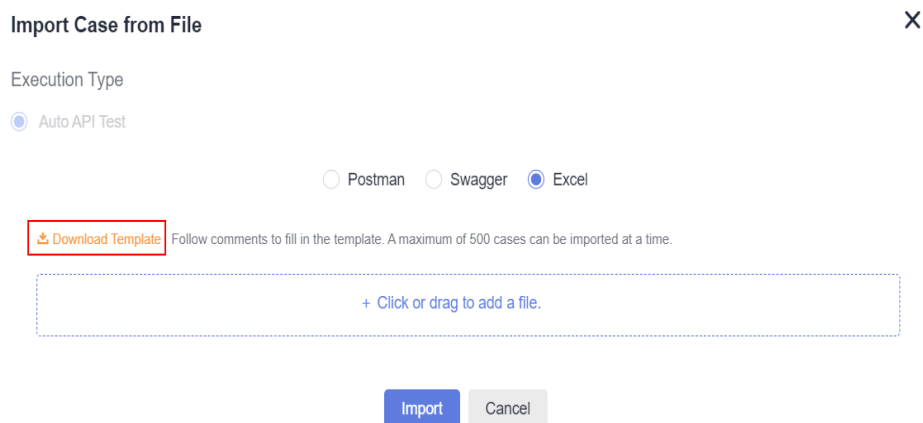
A maximum of 500 test cases can be imported at a time using an Excel file.

**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 choose **Import > Import from File** on the right of the page.

**Step 4** Select **Excel** and click **Download Template**.



- Step 5** 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 fields in the template are as follows.

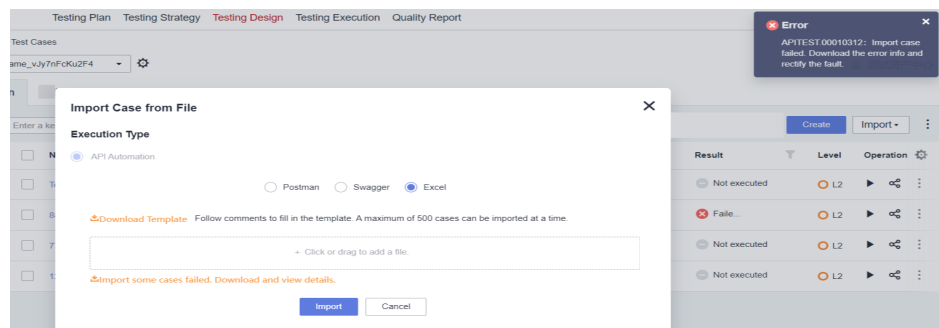
Field	Mandatory	Description
Case Name	Yes	The value can contain only 3 to 128 characters and allows uppercase letters, lowercase letters, digits, and special characters (-_ /*&`'^~:;(){}=+,...—!@#\$\$%.[]<>?-"').
Case Description	No	The value contains no more than 500 characters.
Request Type	Yes	Only GET, POST, PUT, and DELETE are supported.
Request Header Parameter	No	The format is <b>key=value</b> . If there are multiple parameters, separate them with &, that is, <b>key=value&amp;key1=value1</b> .
Request Address	Yes	The request protocol can be HTTP or HTTPS. The format is <b>https://ip:port/pathParam?query=1</b> .
Environment Group	No	Environment parameter group.
IP Variable Name	No	Generates the variable name in the corresponding <b>Environment Group</b> , extracts the content of the <b>Request Address</b> , and generates the corresponding global variable.
Request Body Type	No	The value can be <b>raw</b> , <b>json</b> , or <b>formdata</b> , which corresponds to the text, JSON request body, or form parameter format on the page, respectively. If this parameter is empty, the JSON format is used by default.
Request Body	No	If the request body type is <b>formdata</b> , the request body format is <b>key=value</b> . If there are multiple parameters, separate them with &, that is, <b>key=value&amp;key2=value2</b> . When cases are imported using an Excel file, <b>formdata</b> does not support the request body in file format.
Checkpoint Matching Mode	No	Supports <b>Exact match</b> and <b>Fuzzy match</b> . <b>Exact match</b> indicates <b>Equals</b> , and <b>Fuzzy match</b> indicates <b>Contains</b> .

Field	Mandatory	Description
Expected Checkpoint Value	No	<b>Target Value</b> of the corresponding checkpoint.

**Step 6** 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 7** 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.



In the **Import Case from File** window, you can download the error list, modify the Excel file based on the error causes, and try again.

----End

## 5.4.15 Executing an API Automation Test Case

### Prerequisites

An API automation test case has been created.

### Common Execution

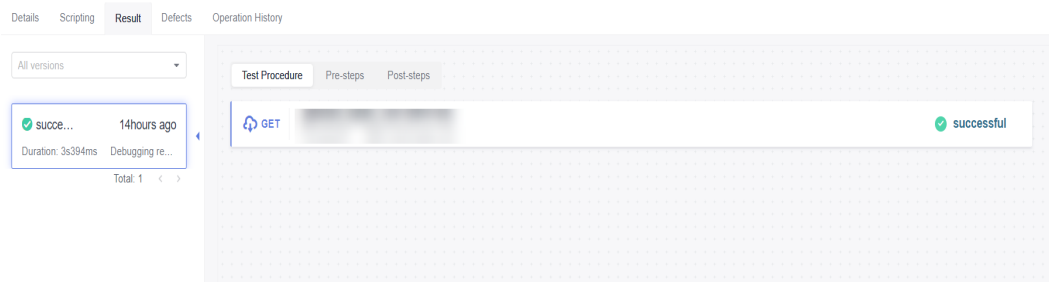
**Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.

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

**Step 3** On the **Auto API Test** tab page, locate the test case to be executed and click ▶ in the **Operation** column to start automatic execution.

**Step 4** After the execution is complete, you can view the execution result in the **Result** column of the test case list.

Click the test case name and click the **Results** tab to view the execution history of the test case.




**Step 5** After confirming the test case execution result, set the test case status to **Completed** on the **Details** tab page.

----End

## Execution with Specified Parameters

API automation 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 dialog box that is displayed, select a version number (version number of the test plan to which the test case belongs) and click **Execute**.

### Specify Parameter to Execute



Version :

Execute

Cancel

----End

## 5.5 Custom Automation Test Cases

### 5.5.1 Creating a Custom Automation Test Case

#### Prerequisites

1. A code repository has been created, and a Python file in .py format has been created in the code repository.

2. An automation execution type has been customized. For details, see [Setting a Custom Execution Type](#).

## Procedure

A custom automation test case consists of basic information and script path.

- Basic information is for managing and describing the test case, including the name (mandatory), ID, tag, test case level, processor, folder, description, prerequisites, test steps, and expected result.
- Script path is the relative path of the custom automation test case in the code repository.

**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 the custom execution mode tab and click **Create** in the upper left corner of the page.

**Step 4** Enter the case name, configure other information as required, set **Script Path** to the relative path of the script file in the code repository, and click **Save**.

### NOTE

Search for the relative path of the created Python file in the code repository, for example, `./scripts/test.py`.

----End

## 5.5.2 Importing a Custom Automation Test Case

### Background

CodeArts TestPlan allows you to generate test cases by importing files in any of the following formats:

- Excel: Excel file based on the given template

### Importing an Excel File

A maximum of 5000 test cases can be imported at a time using an Excel file.

**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 the custom execution mode tab, click **Import** on the right, choose **Import from File** from the drop-down list, and click **Download Template**.

**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 fields in the template are as follows.

Field	Description
Name	Name of a test case (mandatory). Describe the test scenario or function of a test case. The value contains 3 to 128 characters.
ID	ID of a test case. The value contains 3 to 128 characters.
Script Path	Relative path of the script file in the repository.
Processor	Person who needs to complete the test case.
Status	Status of a test case. The value can be <b>New</b> , <b>Designing</b> , <b>Testing</b> , or <b>Completed</b> .
Result	Result of a test case. The value can be <b>Succeeded</b> , <b>Failed</b> , <b>Pending check</b> , <b>Not available</b> , <b>Blocked</b> , or other custom results.
Type	Type of a test case. The value can be <b>Function Test</b> , <b>Performance Test</b> , <b>Compatibility Test</b> , <b>Usability Test</b> , <b>Reliability Test</b> , <b>Security Test</b> , and <b>Serviceability Test</b> .
Level	Test case level based on the importance of the scenario or function. <ul style="list-style-type: none"><li>● <b>L0</b>: verification of underlying functions. Each module should have 10 to 20 test cases. L0 test cases account for 5% of all test cases.</li><li>● <b>L1</b>: verification of basic functions for inherited features or before iteration acceptance. L1 test cases account for 20% of all test cases.</li><li>● <b>L2</b>: verification of important features for manual tests in non-regression versions. L2 test cases account for 60% of all test cases.</li><li>● <b>L3</b>: verification of minor and non-important functions, and exception tests on basic and important functions. L3 test cases account for 10% to 15% of all test cases.</li><li>● <b>L4</b>: verification of special input, scenarios, and threshold conditions. L4 test cases account for less than 5% of all cases.</li></ul>
Iteration	Iteration to test the current test case.
Module	Module to which a test case belongs. The module list comes from the project settings.
Requirement ID	ID of the requirement to be associated with the test case.
Requirement Name	Name of the requirement to be associated with the test case.
Description	Description of the test case.
Prerequisites	Prerequisites for executing the current test case.
Folder	Association with a feature directory.

Field	Description
Test Step	Step description and expected result.
Expected Result	<b>Target Value</b> of the corresponding 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 again.

----End

## 5.6 Advanced Configurations of API Automation

### 5.6.1 Built-in Functions

#### 5.6.1.1 Binary Addition Operation

##### Function Name

\$add(intA, intB)

##### Function Description

Implements the addition operation between parameter A and parameter B, which support the following types:

- Numbers
- Environment parameters
- 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**.

Request Information:			
GET	https://	?test=\$add(1000,1000)	URL Parameter
Name	Value	Description	Operation
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**.

Request Header	Value	Description	Operation
Content-Type	application/json		🗑️ +
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 Type	Type Value	Comparison Operator	Target Value	Operation
ody (JSON)	result	N/A		==	\$add(\${test},1)	🗑️ ↓ ↑ +

- **if** condition

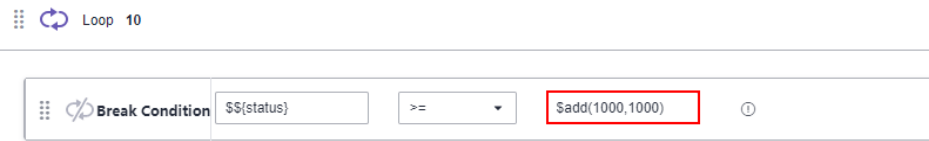
As shown in the following figure, the target value of the **if** condition is the binary addition function. Parameter A in the function is **1**, and parameter B is the environment variable **status**. For details about how to set environment parameters, see [5.4.13 Setting Environment Parameters](#).

IF	Condition	Operator	Target Value
IF	\$(test)	<=	\$add(1,\$S(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**.





## 5.6.1.2 Binary Subtraction Operation

### Function Name

\$subtract(intA, intB)

### Function Description

Implements the subtraction operation between parameter A and parameter B, which support the following types:

- Numbers
- Environment parameters
- Local parameters
- Binary operations

### Application Scenarios

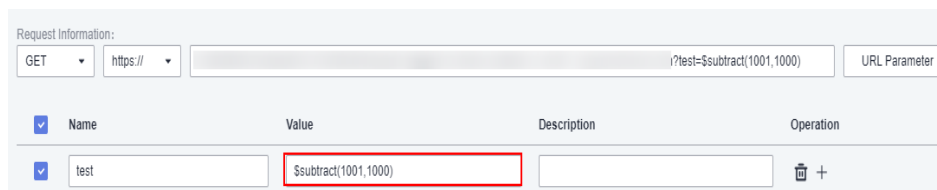
The binary subtraction function can be used in the following scenarios for API automation:

- 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.



- 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**.

Request Header	Value	Description	Operation
Content-Type	application/json		🗑️ +
subtract	<b>\$subtract(\$multiply(100,100),-1)</b>		🗑️ +

- 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)**.

```
Request Body:
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 }
7
8 }
```

- 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 Type	Type Value	Comparison Operator	Target Value	Operation
e body (JSON)	result			==	<b>\$multiply(\$test,1)</b>	🗑️ ↓ ↑ +

- 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 [5.4.13 Setting Environment Parameters](#).

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.

Loop 10	Break Condition	\$status	>	<b>\$subtract(1001,1000)</b>
---------	-----------------	----------	---	------------------------------

### 5.6.1.3 Binary Multiplication Operation

#### Function Name

\$multiply(intA, intB)

#### Function Description

Implements the multiplication operation between parameter A and parameter B, which support the following types:

- Numbers
- Environment parameters
- 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	Description	Operation
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**.

Request Header	Value	Description	Operation
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)**.

```
Request Body:
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
7
8
```

- 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](#).

Source	Property	Advanced Extraction Type	Type Value	Comparison Operator	Target Value	Operation
e body (JSON)	result			==	\$multiply(\$test,1)	🗑️ ⬆️ ⬇️ ⬆️ ⬆️

- 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 [5.4.13 Setting Environment Parameters](#).

IF	Condition	Operator	Target Value
\$test	<=	\$multiply(100,\$status)	

- 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**.

Break Condition	Condition	Operator	Target Value
SS(status)	>	\$multiply(100,100)	

### 5.6.1.4 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
- Environment parameters
- 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**.

Request Information:

GET https:// ?test=Sdivide(1000,100) URL Parameter

Name	Value	Description	Operation
test	Sdivide(1000,100)		+

- 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**.

Request Information:

GET https:// ?test=Sdivide(1,3,5) URL Parameter

Name	Value	Description	Operation
test	Sdivide(1,3,5)		+

- 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**.

Request Header	Value	Description	Operation
Content-Type	application/json		🗑️ +
add	<b>\$add(\$subtract(1001,1000),-1)</b>		🗑️ +

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**.

Request Header	Value	Description	Operation
divide	<b>Sdivide(1,-3,\$\$scale)</b>		🗑️ +

- 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)**.

```
Request Body:
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 }
7
8
9
```

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**.

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

- 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](#).

Source	Property	Advanced Extraction Type	Type Value	Comparison Operator	Target Value	Operation
e body (JSON)	result			==	<b>Sdivide(\$test,1)</b>	🗑️ ↓ ↑ +

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](#).

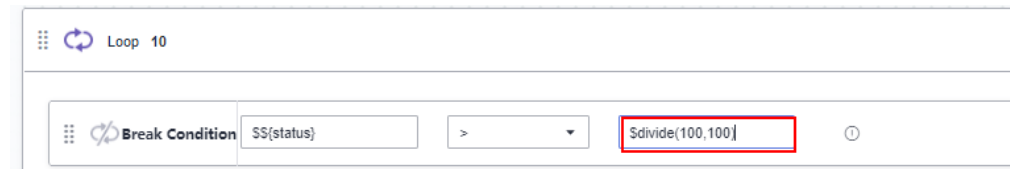


As shown in the following figure, the target value of the **if** condition is the binary division function with a precision value. Parameter A in the function is **1** and parameter B is **3**. For details about how to set environment parameters, see [5.4.13 Setting Environment Parameters](#).

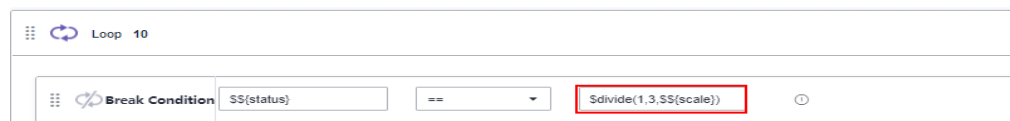


- **for loop interrupt condition**

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 **SS{scale}**.



### 5.6.1.5 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

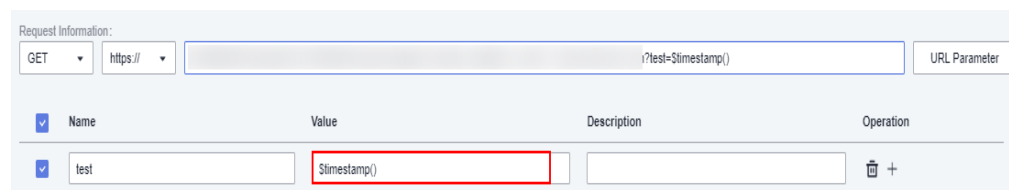
The function for obtaining the current timestamp can be used in the following scenarios for API automation:

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

## Example

- Request URL

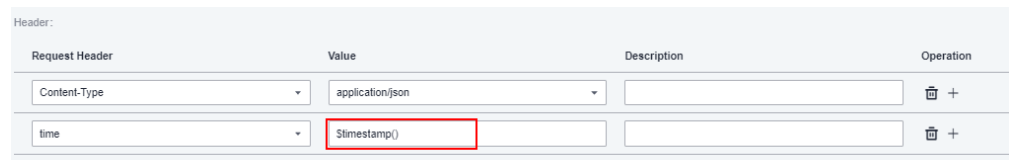
As shown in the following figure, the value of **test** in the request URL is the function for generating the current timestamp.



The screenshot shows the 'Request Information' configuration panel. The 'Request Method' is set to 'GET' and the 'URL' is 'https://' followed by a text input field containing '/?test=\$timestamp()'. Below this is a table with columns: Name, Value, Description, and Operation. A row is added with 'test' in the Name column and '\$timestamp()' in the Value column, which is highlighted with a red box.

- Request header

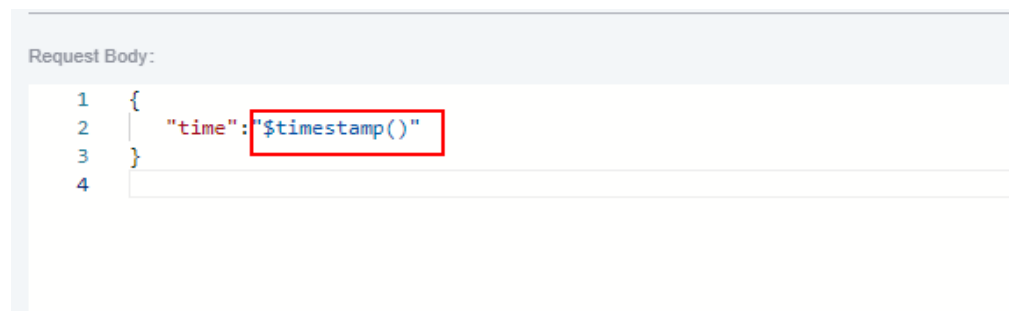
As shown in the following figure, the value of **time** in the request header is the function for generating the current timestamp.



The screenshot shows the 'Request Header' configuration panel. It contains a table with columns: Request Header, Value, Description, and Operation. Two rows are present: one for 'Content-Type' with value 'application/json', and another for 'time' with value '\$timestamp()', which is highlighted with a red box.

- Request body

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



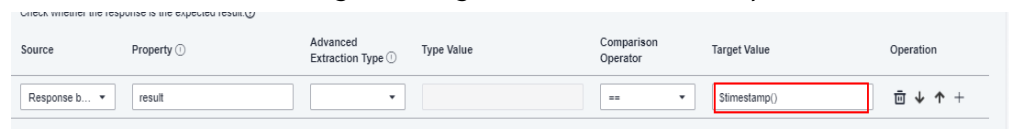
The screenshot shows the 'Request Body' configuration panel. It displays a JSON body with the following content:

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

The value '\$timestamp()' is highlighted with a red box.

- Checkpoint property

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

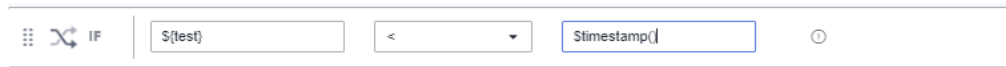


The screenshot shows the 'Checkpoint property' configuration panel. It contains a table with columns: Source, Property, Advanced Extraction Type, Type Value, Comparison Operator, Target Value, and Operation. A row is added with 'Response b...' in the Source column, 'result' in the Property column, and '\$timestamp()' in the Target Value column, which is highlighted with a red box.



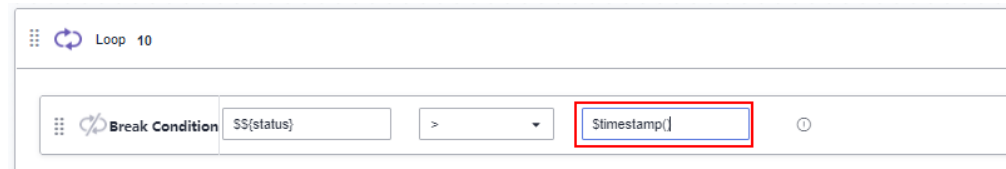
- **if** condition

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



### 5.6.1.6 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
- Environment parameters
- Local parameters
- Other built-in functions

#### Application Scenarios

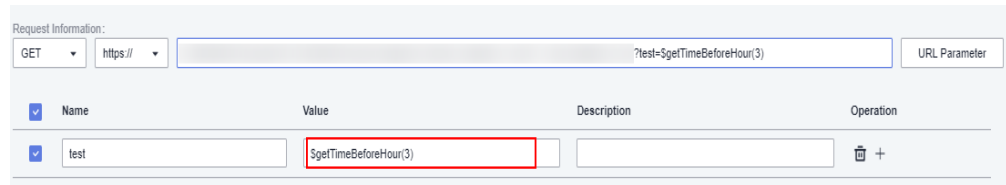
The function for obtaining the specified timestamp can be used in the following scenarios for API automation:

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

#### Example

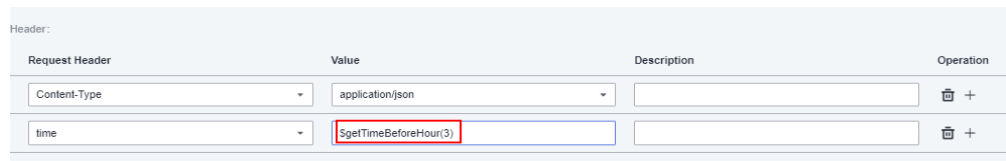
- Request URL

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**.



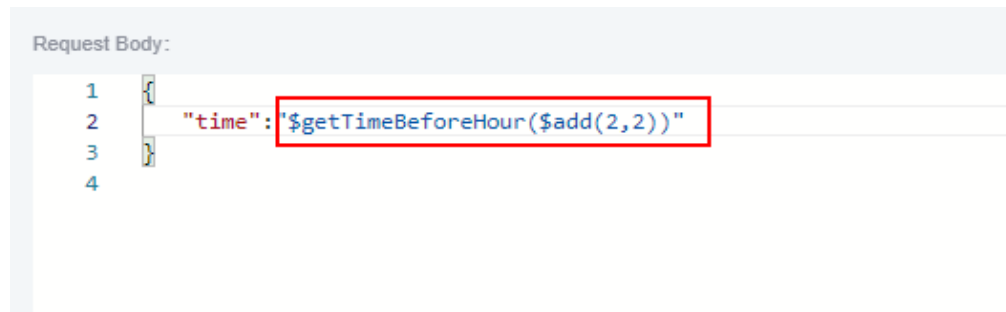
- Request header

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**.



- 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)**.



- 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](#).



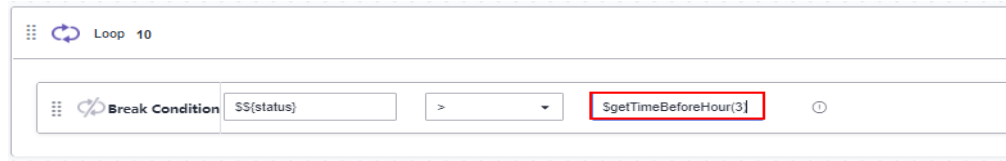
- if condition

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 [5.4.13 Setting Environment Parameters](#).



- 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**.



### 5.6.1.7 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](#)
- Environment parameters
- 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 [5.4.13 Setting Environment Parameters](#).

Name	Value	Description	Operation
test	SdateFormat(\$\$(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**.

Request Header	Value	Description	Operation
Content-Type	application/json		🗑️ +
time	SdateFormat(\$\$(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": "SdateFormat(2020.09.11 11:00:00)"
3 }
4
```

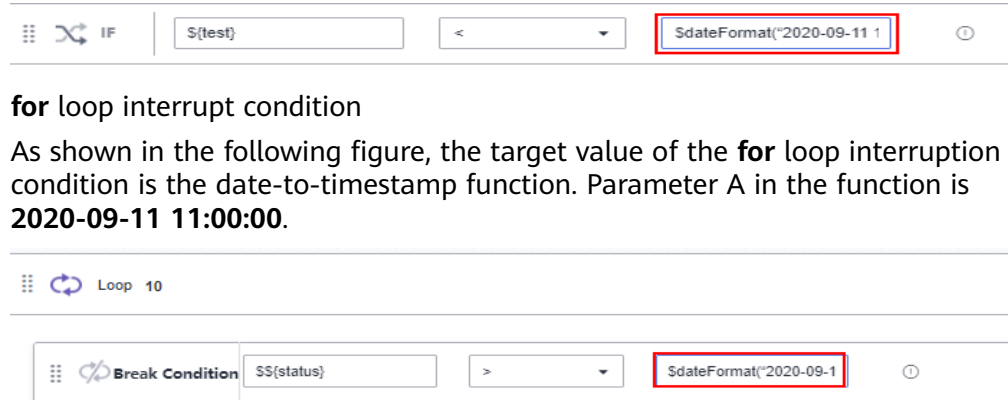
- Checkpoint property

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 Type	Type Value	Comparison Operator	Target Value	Operation
Respo...	result			==	SdateFormat(\$\$(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**.

### 5.6.1.8 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.

#### 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**.

In a date and time value, spaces, hyphens (-), slashes (/), and colons (:) are optional. For example, the 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

## 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

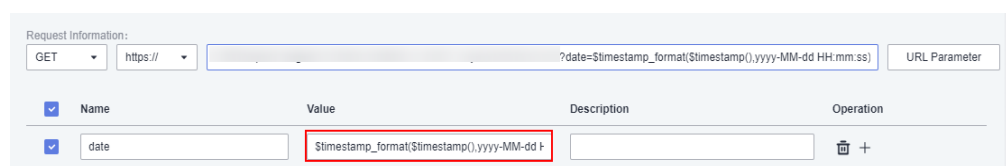
The timestamp-to-date function can be used in the following scenarios for API automation:

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

## Example

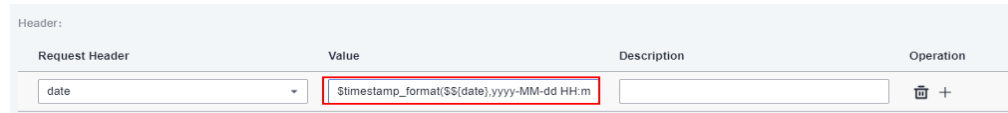
- Request URL

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 [5.6.1.5 Obtaining the Current Timestamp](#). Parameter B is **yyyy-MM-dd HH:mm:ss**.



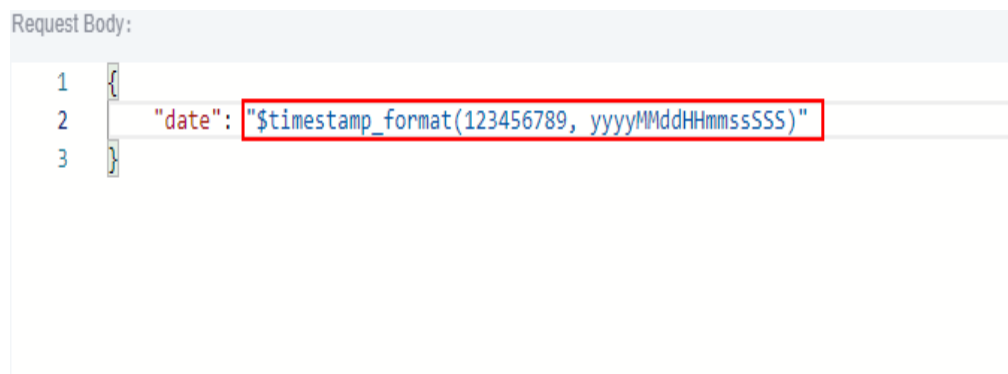
- Request header

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 [5.4.13 Setting Environment Parameters](#).



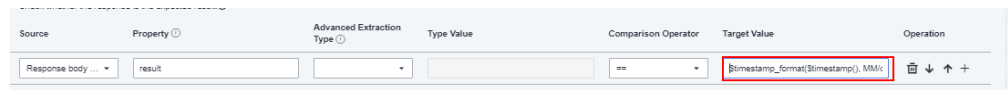
- Request body

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**.



- 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**.



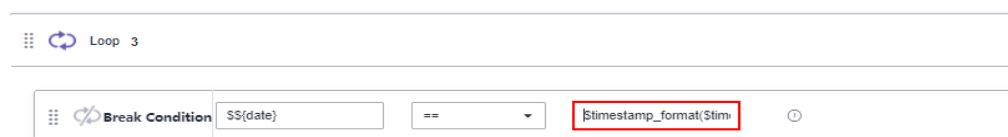
- if condition

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**.



## 5.6.1.9 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](#)
- Environment parameters
- 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	Description	Operation
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**.

Request Header	Value	Description	Operation
Content-Type	application/json		+
time	\$timeStampCalculation(\$timestamp(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**.

```
Request Body:
1 {
2   "time": "$timeStampCalculation($dateFormat(2020.09.11 11:00:00),1d)"
3 }
4
5
```

- 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 [5.4.13 Setting Environment Parameters](#).

Source	Property	Advanced Extraction Type	Type Value	Comparison Operator	Target Value	Operation
Response body ...	result			==	\$timeStampCalculation(\$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 [5.4.13 Setting Environment Parameters](#).

IF	Condition	Operator	Target Value
\$test	<=	\$timeStampCalculation(\$status,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**.

Break Condition	Condition	Operator	Target Value
\$status	>=	\$timeStampCalculation(1607939485441,1d)	

## 5.6.1.10 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

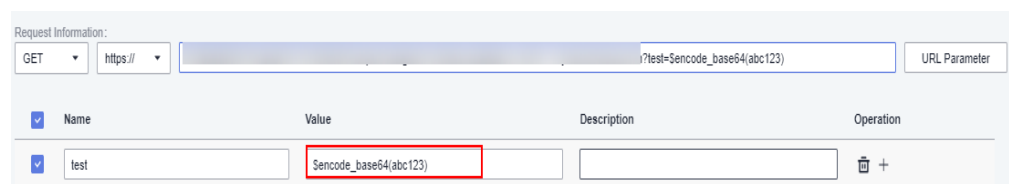
The Base64 encoding generation function can be used in the following scenarios for API automation:

- 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**.

Request Header	Value	Description	Operation
Content-Type	application/json		+
Accept-Encoding	Sencode_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()**.

```
Request Body:
1  {
2    "encoding": "$encode_base64($uuid())"
3  }
4
5
```

- 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 Type	Type Value	Comparison Operator	Target Value	Operation
Responses...	result			Contains	Sencode_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 [5.4.13 Setting Environment Parameters](#).

IF	\$(test)	Contains	Sencode_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**.

Loop 10			
Break Condition	\$\$status	Does not contain	Sencode_base64(abc123)

### 5.6.1.11 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	Description	Operation
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**.

Request Header	Value	Description	Operation
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()**.

```
Request Body:
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](#).

Check whether the response is the expected result ⓘ

Source	Property ⓘ	Advanced Extraction Type ⓘ	Type Value	Comparison Operator	Target Value	Operation
Respons...	result			Contains	\$sha512(\$test)	🗑️ ↓ ↑ +

- 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 [5.4.13 Setting Environment Parameters](#).

⋮	IF	\$(test)	Contains	\$sha512(\$status)	ⓘ
---	----	----------	----------	--------------------	---

- 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**.

⋮	Loop 10				
⋮	Break Condition	\$(status)	Does not ...	\$sha512(abc123)	ⓘ

### 5.6.1.12 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**.

The screenshot shows a 'Request Information' section with a dropdown menu set to 'GET' and a URL field containing 'https://' followed by a parameter box with the value '?test=\$md5(abc123)'. Below this is a table with columns for Name, Value, Description, and Operation. The table contains one row with 'test' in the Name column and '\$md5(abc123)' in the Value column. The Value cell is highlighted with a red border.

Name	Value	Description	Operation
test	\$md5(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**.

The screenshot shows a 'Header' section with a table for request headers. The table has columns for Request Header, Value, Description, and Operation. The second row has 'Accept-Encoding' in the Request Header column and '\$md5(abc123)' in the Value column.

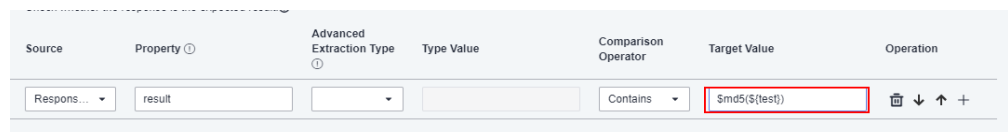
Request Header	Value	Description	Operation
Content-Type	application/json		🗑️ +
Accept-Encoding	\$md5(abc123)		🗑️ +

- Request body

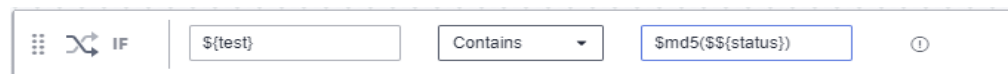
As shown in the following figure, the request body uses the MD5 hash value generation function. Parameter A in the function is **\$uuid()**.

```
Request Body:
1  {
2    "encoding": "$md5($uuid())"
3  }
4
5
```

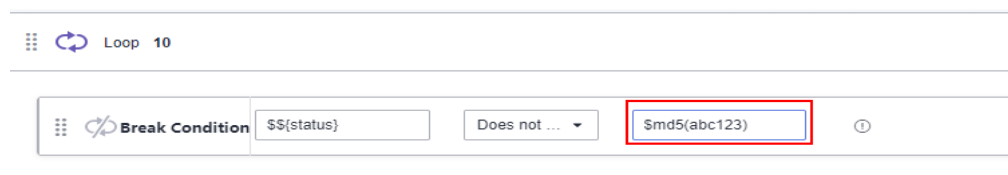
- **Checkpoint property**  
As shown in the following figure, the target value of the checkpoint property **result** is the MD5 hash value generation function. Parameter A in the function is the local parameter **test**. For details about how to set local parameters, see [Local Parameters](#).



- **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 [5.4.13 Setting Environment Parameters](#).



- **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**.



### 5.6.1.13 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**.

Request Information:			
Name	Value	Description	Operation
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**.

Header:			
Request Header	Value	Description	Operation
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)**.



```
Request Body:
1  {
2    "number": "$random_int($add(5,5),$multiply(5,5))"
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 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](#).

Source	Property	Advanced Extraction Type	Type Value	Comparison Operator	Target Value	Operation
Respons...	result			==	\$random_int(1,\$(test))	🗑️ ↓ ↑ +

- 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 [5.4.13 Setting Environment Parameters](#).

IF	\$(test)	==	\$random_int(1,\$\$(status))
----	----------	----	------------------------------

- 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**.

Loop 10	Break Condition	\$\$\$(status)	!=	\$random_int(1,100)
---------	-----------------	----------------	----	---------------------

### 5.6.1.14 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**.

Request Information:

Name	Value	Description	Operation
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**.

Header:

Request Header	Value	Description	Operation
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)**.



- 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	Description	Operation
test	\$uuid()		

- Request header

As shown in the following figure, the value of **time** in the request header is the UUID generation function.

Request Header	Value	Description	Operation
time	\$uuid()		

- Request body

As shown in the following figure, the request body uses the UUID generation function.

```

1 {
2   "string": "$uuid()"
3 }
    
```

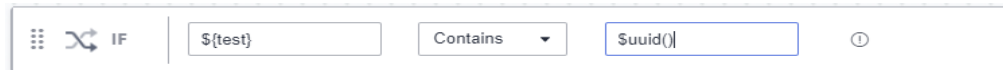
- Checkpoint property

As shown in the following figure, the target value of the checkpoint property **result** is the UUID generation function.

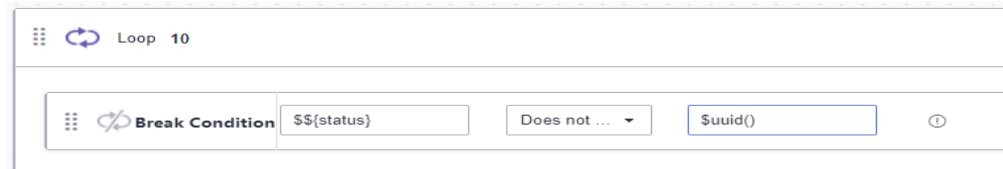
Source	Property	Advanced Extraction Type	Type Value	Comparison Operator	Target Value	Operation
Respo...	result			Contains	\$uuid()	

- **if** condition

As shown in the following figure, the target value of the **if** condition is the UUID generation function.



- **for** loop interrupt condition  
As shown in the following figure, the target value of the **for** loop interrupt condition is the UUID generation function.



### 5.6.1.16 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

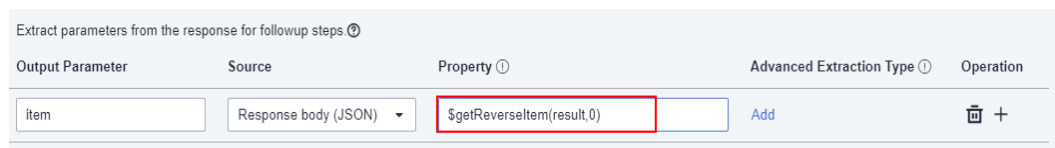
Obtains an array from a response body or header based on the reverse index.

#### 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**.



### 5.6.1.17 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	Advanced Extraction Type	Operation
item	Response body (JSON)	<code>\$getReverseItem(result,0)</code>	Add	+

### 5.6.1.18 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
- Environment parameters
- 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**.

The screenshot shows the 'Request Information' dialog box. The 'Name' field contains 'test' and the 'Value' field contains '\$toLower(TEST)'. The 'URL Parameter' checkbox is checked. The 'Operation' column shows a trash icon and a plus sign.

- 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**.

The screenshot shows the 'Header' dialog box. The 'Request Header' field contains 'low' and the 'Value' field contains '\$toLower(Test)'. The 'Operation' column shows a trash icon and a plus sign.

- 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**.

The screenshot shows the 'Request Body' dialog box. The JSON body is:
 

```

    {
      "id": 1,
      "lower": "$toLower(TEST)"
    }
    
```

 The value '\$toLower(TEST)' is highlighted with a red box.

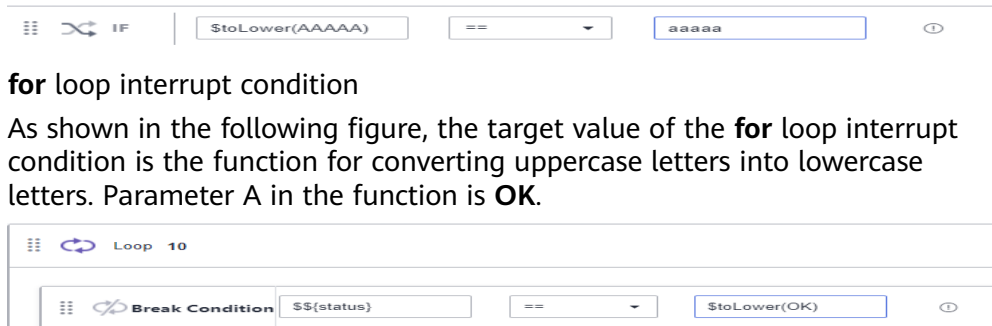
- Checkpoint property

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**.

The screenshot shows the 'Check whether the response is the expected result' dialog box. The 'Property' field contains 'result' and the 'Target Value' field contains '\$toLower(Test)'. The 'Comparison Operator' is set to '=='. The 'Operation' column shows a trash icon, a down arrow, an up arrow, and a plus sign.

- **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**.



- **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**.

### 5.6.1.19 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
- Environment parameters
- Local parameters

#### Function Description

Converts all characters in the specified input string into uppercase characters.

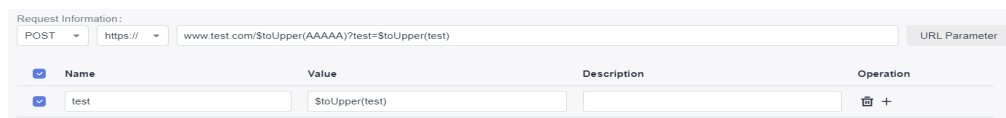
#### Application Scenarios

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

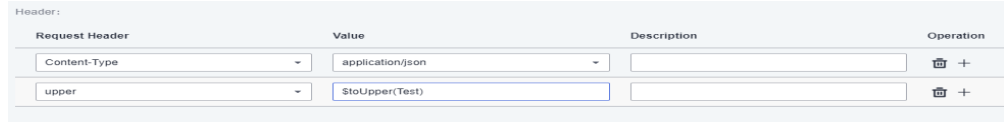
- Request URL  
As shown in the following figure, the value of **test** in the request URL is the function for converting lowercase letters into uppercase letters. Parameter A in the function is **test**.





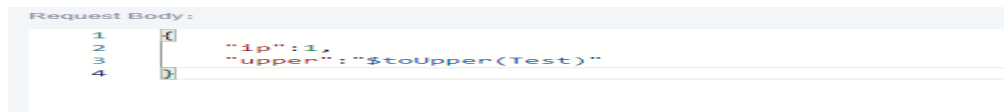
- Request header

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**.



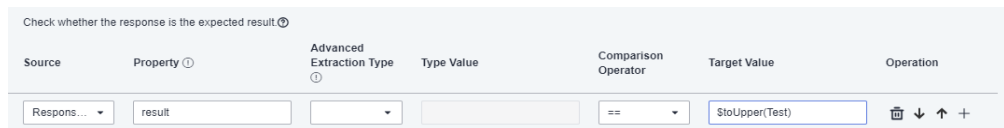
- Request body

As shown in the following figure, the function for converting lowercase letters into uppercase letters is used. Parameter A in the function is **Test**.



- 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**.



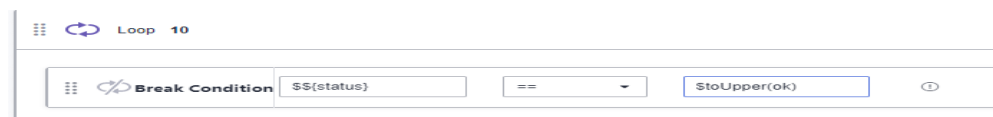
- if condition

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**.



- 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**.



### 5.6.1.20 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
- Environment parameters
- Local parameters

## Function Description

Concatenates strings 1 and 2 into a new string.

## Application Scenarios

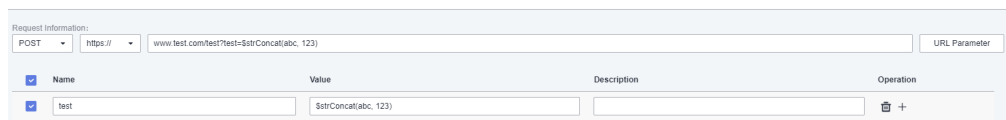
The string concatenation function can be used in the following scenarios for API automation:

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

## Example

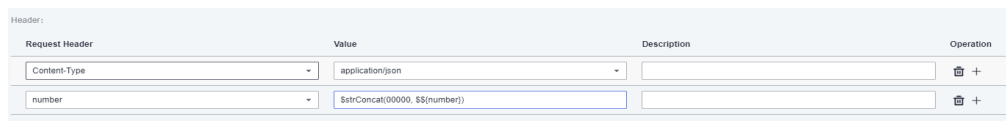
- Request URL

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.



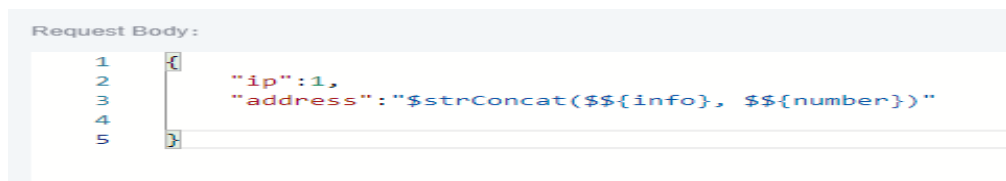
- Request header

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}`**.



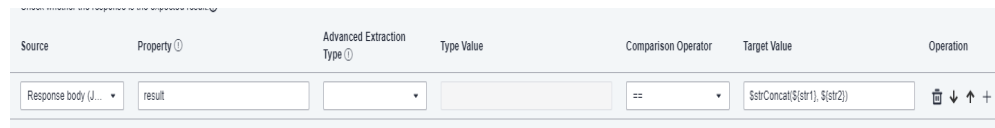
- Request body

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}`**.



- 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}`**.



- **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**.



### 5.6.1.21 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
- Environment parameters
- 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**.

The screenshot shows the 'Request Information' dialog box. The 'Request Information' section includes a dropdown for 'POST', a 'https://' protocol, and a URL: 'www.test.com/test?name=\${strCut}(\$\${user}, 2, 4)'. Below this is a table with columns: Name, Value, Description, and Operation. A single row is visible with 'name' in the Name column, '\${strCut}(\$\${user}, 2, 4)' in the Value column, and an empty Description and Operation column.

- 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**.

The screenshot shows the 'Header' dialog box. It contains a table with columns: Request Header, Value, Description, and Operation. Two rows are visible: the first row has 'number' in the Request Header column and '\${strCut}(\$\${user}, 2, 4)' in the Value column; the second row has 'Content-Type' in the Request Header column and 'application/json' in the Value column.

- 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**.

The screenshot shows the 'Request Body' dialog box. It displays a JSON configuration with line numbers 1 through 4. The JSON content is: 

```
{
  "ip": "${toUpper}($${a})",
  "name": "${strCut}($${user}, 2, 4)"
}
```

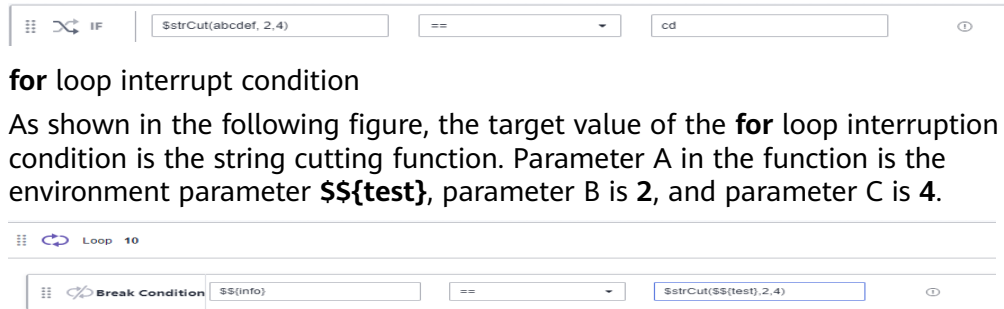
- Checkpoint property

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**.

The screenshot shows the 'Checkpoint Property' dialog box. It has a table with columns: Source, Property, Advanced Extraction Type, Type Value, Comparison Operator, Target Value, and Operation. A single row is visible with 'Response body (J...)' in the Source column, 'result' in the Property column, an empty Advanced Extraction Type column, an empty Type Value column, '==' in the Comparison Operator column, and '\${strCut}(\$\${info}, 2, 4)' in the Target Value column.

- **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**.



- **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**.

## 5.6.2 Advanced Extraction Types

### 5.6.2.1 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:

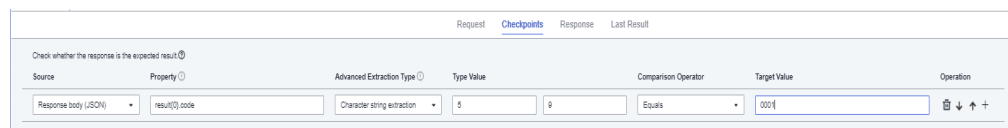
- The first box indicates the start index. The first digit is **0**. The start index is contained in the truncated string.
- The second box indicates the end index. The end index is not contained in the truncated string.

#### Example

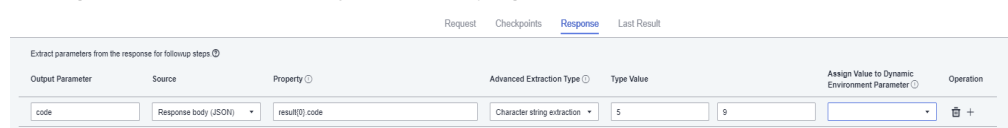
The following response body is used as an example.



- 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** tab page as follows.



- 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 **Extract Response** tab page as follows.



### 5.6.2.2 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**.

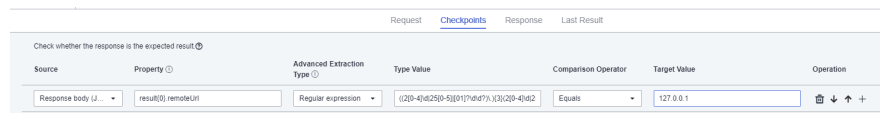
The Java regular expression engine can be used.

## Example

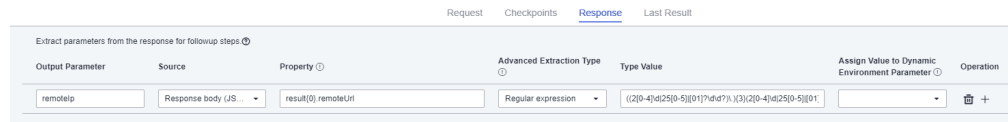
The following response body is used as an example.



- If the checkpoint verifies **remoteUrl** of the first element in the **result** array in the response body, obtain the IP value through regular expression matching and compare the value with the target value. The configuration is as follows. Set **Type Value** to `((2[0-4]\d|25[0-5]|\d{1,3})\d{0,2})\.\{3\}(2[0-4]\d|25[0-5]|\d{1,3})\d{0,2})`.



- If **remoteUrl** of the first element in the **result** array in the response body is extracted, obtain the IP value through regular expression matching and assign the value to the variable **remotelp**. The configuration is as follows. Set **Type Value** to `((2[0-4]\d|25[0-5]|\d{1,3})\d{0,2})\.\{3\}(2[0-4]\d|25[0-5]|\d{1,3})\d{0,2})`.



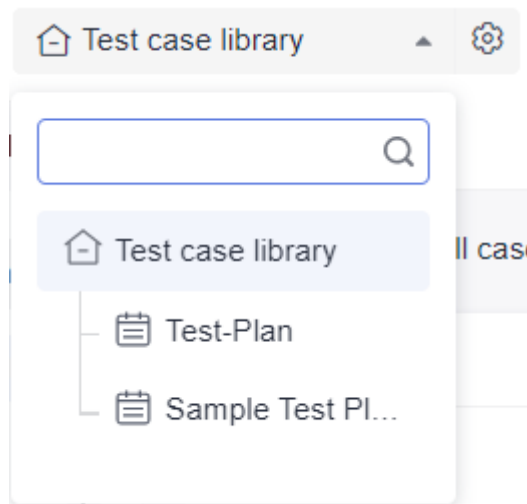
## 5.7 Adding Test Cases in Batches

### Background

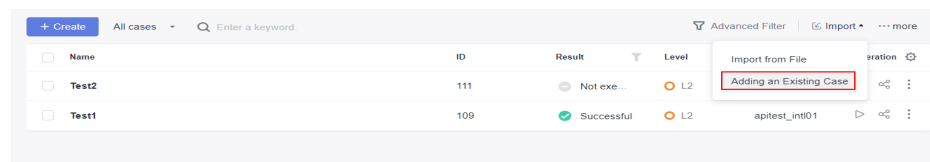
CodeArts TestPlan allows you to add test cases to test plans in batches from the test case library, including manual test cases and API automation test cases.

### Adding Manual Test Cases 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** On the **Manual Test** tab page, click **Import** in the right area and select **Adding an Existing Case** 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



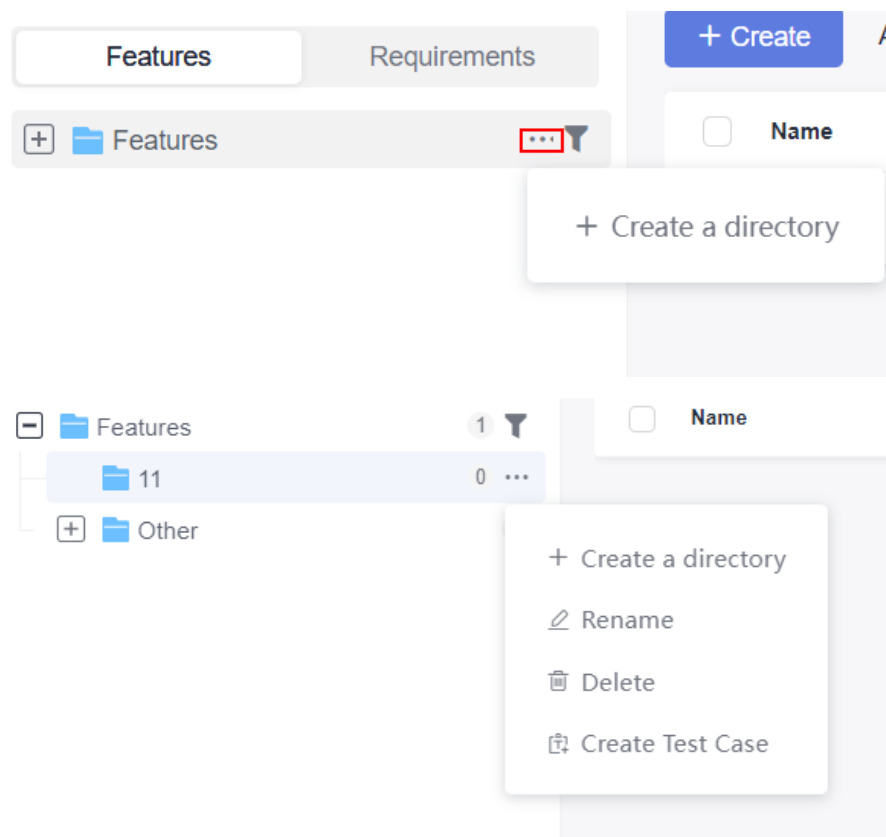
## Adding API Automation Test Cases in Batches

On the **Testing Case** page, click the **Auto API Test** tab to add API automation test cases in batches. The method is the same as that in [Adding Manual Test Cases in Batches](#).

## 5.8 Managing Test Cases in the Features Directory

### 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** Click **Features** on the left of the page.
  - By default, all test cases belong to this **Features** directory, which has a default subdirectory **Other**. Click **...** next to the **Features** directory to create another subdirectory. Click **...** next to the new subdirectory to delete or rename the subdirectory. You can also create test cases or subdirectories in the new subdirectory.



----End



## 5.9 Test Cases and Requirements

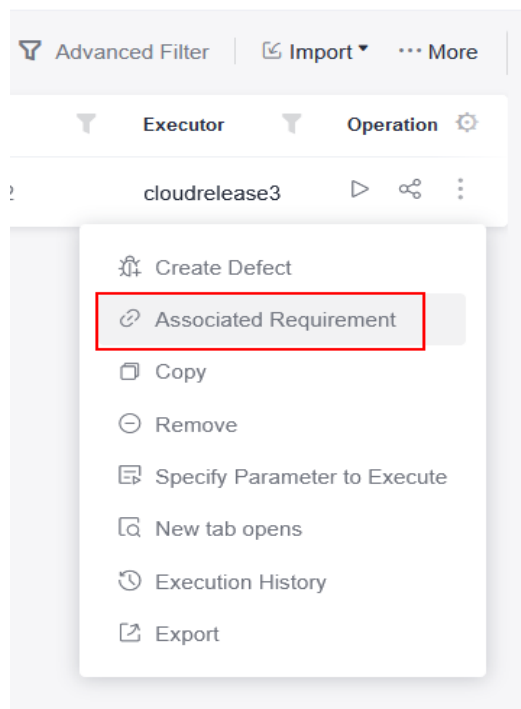
### Associating Test Cases with Requirements

 NOTE

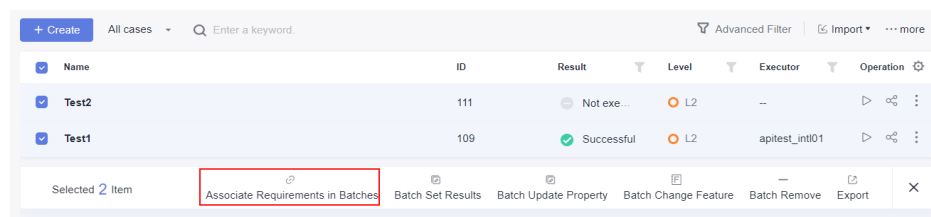
Test cases can be associated only with **Epic**, **Feature**, and **Story** work items of a Scrum project.

CodeArts TestPlan supports association between test cases and requirements. The procedure is as follows:

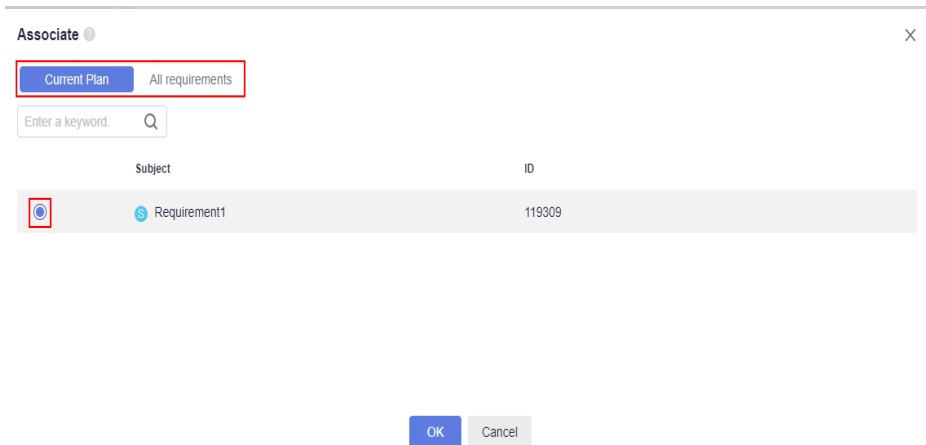
- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** Select a test case to be associated with a requirement, click  on the right, and select **Associated Requirement**. Alternatively, on the **All Cases** tab page, click the  icon in the **Operation** column of the case to be associated with a requirement.



To associate multiple test cases with one requirement, select the required test cases in the list and click **Associate Requirements in Batches** at the bottom of the page.



**Step 4** In the displayed dialog box, select the requirement to be associated. You can select a requirement on the **Current Plan** or **All requirements** tab page. Click **OK**.



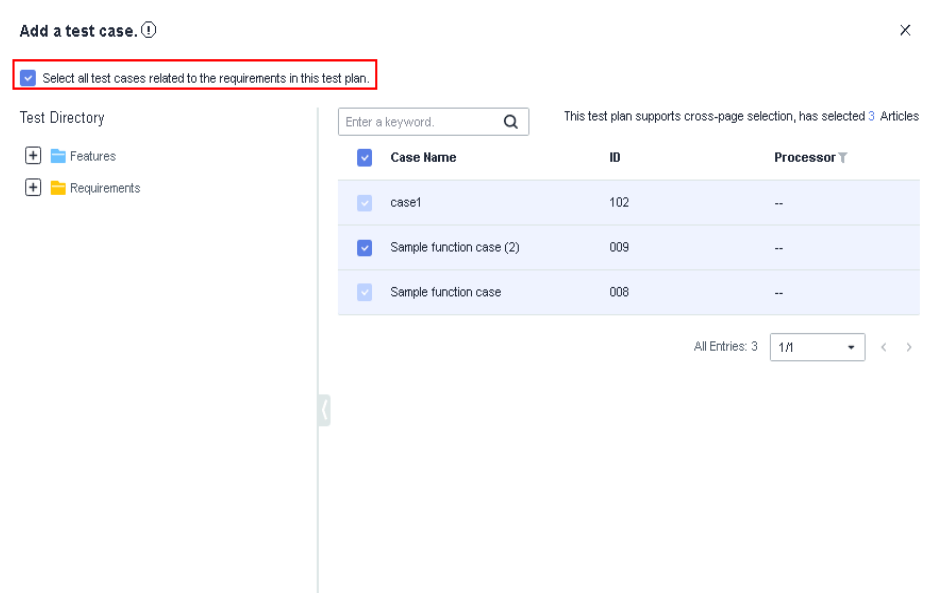
----End

## Adding Test Cases Related to Requirements

- **Prerequisites**

The requirements in the test plan have been associated with test cases in the test case library.


The procedure for adding test cases related to requirements is the same as that in [5.7 Adding Test Cases in Batches](#). In the dialog box that is displayed, select the **Select all test cases related to the requirements in this test plan** check box.

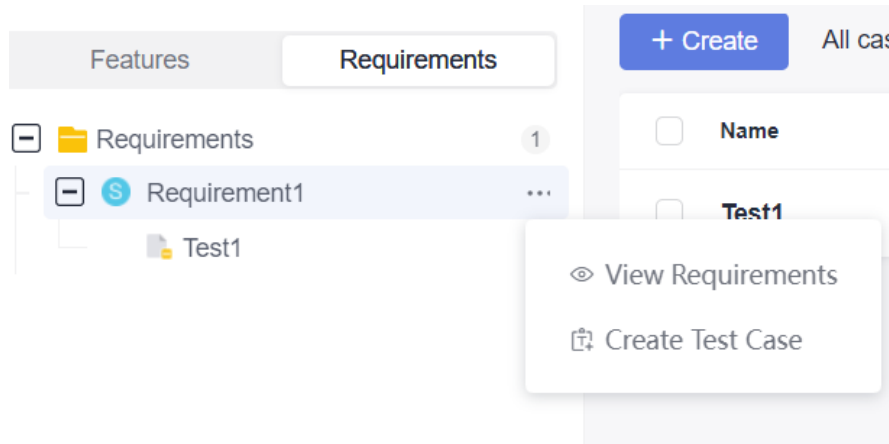


## Managing Test Cases Based on Requirements

On the **Testing Case** page, click the **Requirements** directory in the left pane.

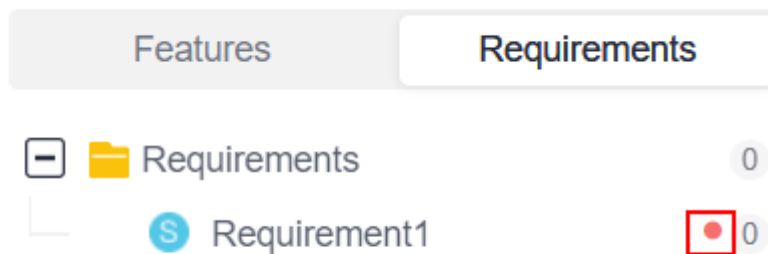
- By default, all associated requirements belong to the **Requirements** directory.

- Click a requirement in the **Requirements** directory to view all cases associated with the requirement.
- Click  on the right of the requirement name to view the requirement details or create a test case associated with the requirement.



## Setting Requirement Change Notification

When a requirement is associated with a test case and the requirement is modified in CodeArts Req, a red dot is displayed after the requirement name on the **Testing Case** page. You need to supplement or modify the test case associated with the requirement.



## 5.10 Test Cases and Defects

When a test case fails to be executed, the test case is usually associated with a defect. You can create a defect or associate the test case with an existing defect.


The following uses manual test cases as an example.

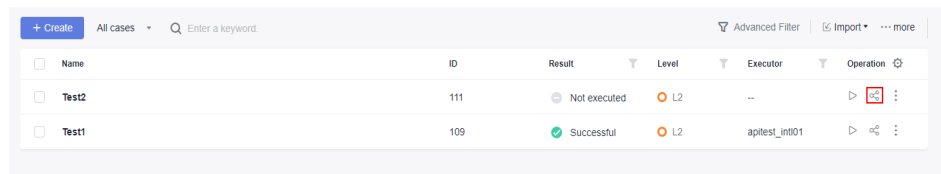
### NOTE



Test cases can be associated only with **Bug** work items in a Scrum project.


## Procedure

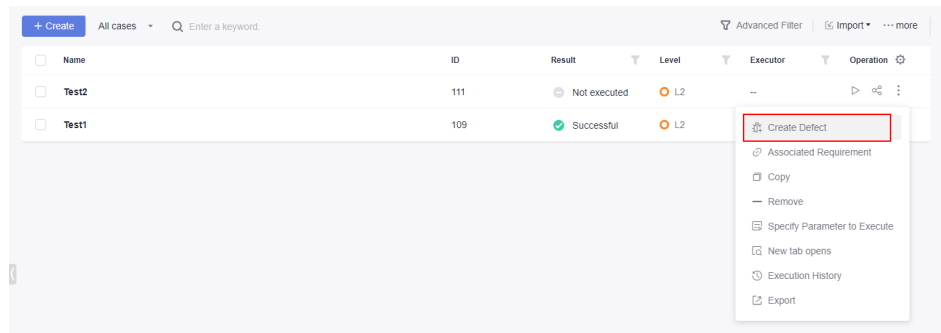
- Step 1** Log in to the CodeArts homepage, search for your target project, and click the project name to access the project.
- Step 2** In the navigation pane, choose **Testing > Testing Case**.
- Step 3** Select a test case to be associated with a defect. You can create or associate a defect using either of the following methods:
- Perform the following operations on the **Testing Case** page:


- Click  in the **Operation** column to associate an existing defect in the current project.




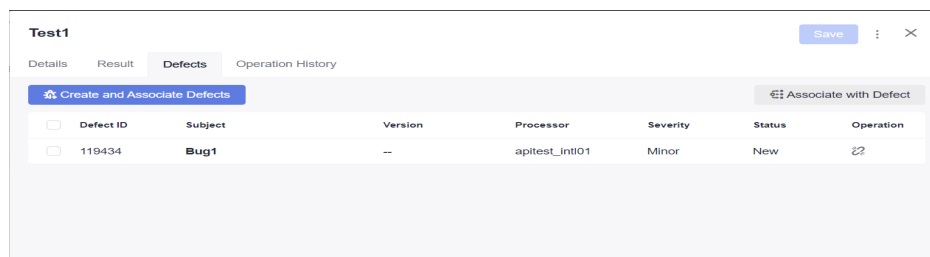
Name	ID	Result	Level	Executor	Operation
Test2	111	Not executed	L2	--	
Test1	109	Successful	L2	apitest_intl01	


- Click  in the **Operation** column, and select **Create Defect** to create a defect as prompted.



Name	ID	Result	Level	Executor	Operation
Test2	111	Not executed	L2	--	 Create Defect Associated Requirement Copy Remove Specify Parameter to Execute New tab opens Execution History Export
Test1	109	Successful	L2		

- Step 4** After a defect is created or associated, view the defect information on the **Defects** tab page. You can click  to disassociate the current defect.



Defect ID	Subject	Version	Processor	Severity	Status	Operation
119434	Bug1	--	apitest_intl01	Minor	New	

----End

## 5.11 Commenting on Test Cases

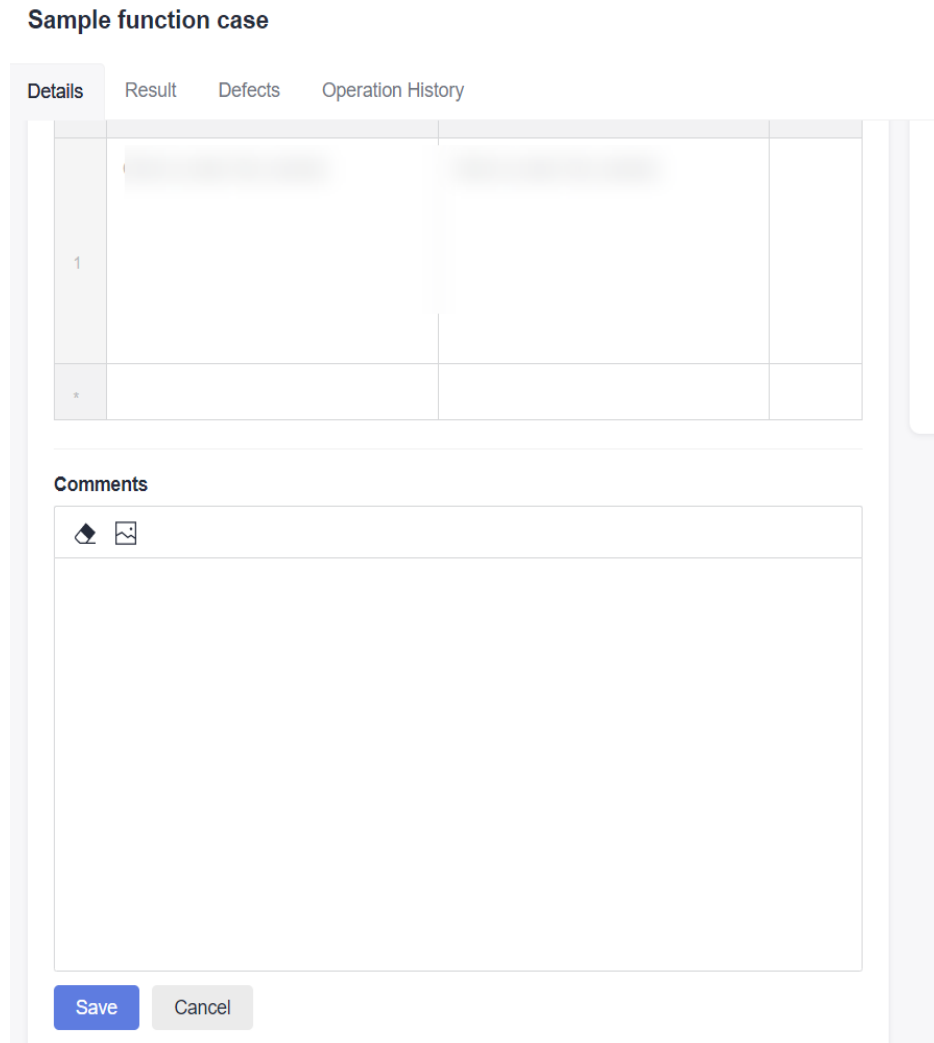
CodeArts TestPlan allows you to comment on test cases.

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

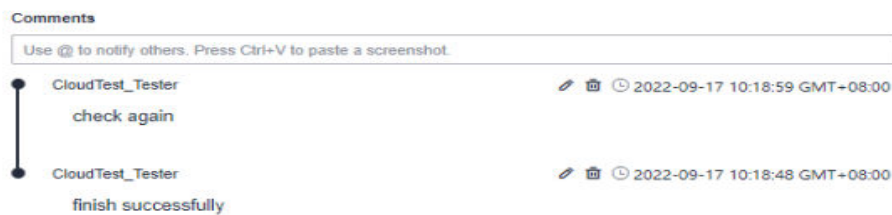
**Step 2** In the navigation pane, choose **Testing > Testing Case**.

**Step 3** Select a test case to be commented on, click the test case name, and click the **Details** tab.

**Step 4** 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

## 5.12 Filtering Test Cases

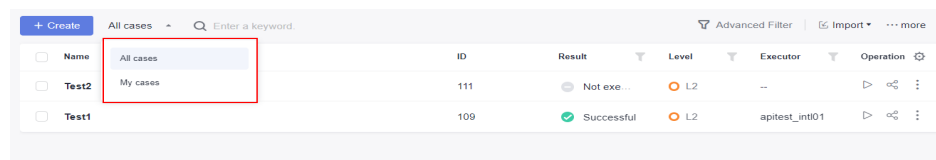
CodeArts TestPlan supports custom filtering of test cases. The following procedure uses the **Testing Case > Manual Test** as an example.

### Using the Default Filter Criteria

- 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 **Manual Test** tab page, select an option from the **All cases** drop-down list.

There are two default filter criteria: **All cases** and **My cases**.

- **All cases:** displays all cases in the current test plan or case library.
- **My cases:** displays all cases whose **Executor** is the current login user.

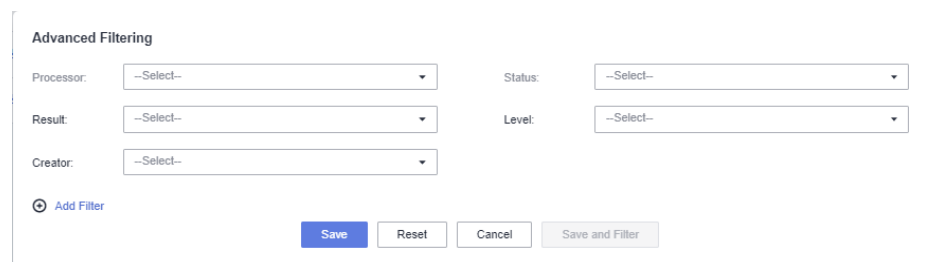


----End

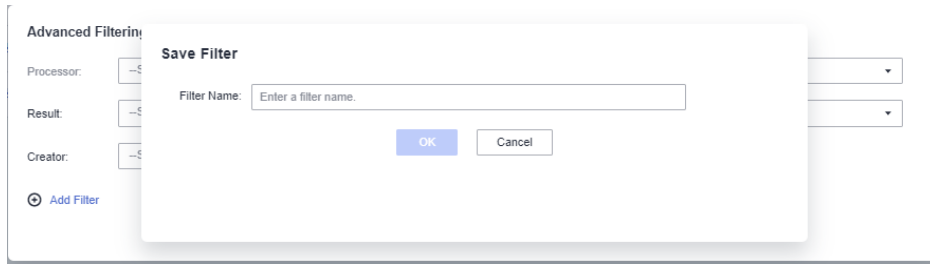
### Setting Advanced Filter Criteria

If the default filter criteria do not meet your requirements, you can customize filter criteria.

- 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 **Advanced Filterer** above the test case list. The **Processor**, **Status**, **Level**, **Creator**, and **Result** filters are displayed:
- Step 4** Set filter criteria as required and click **Save**. Test cases that meet the filter criteria are displayed on the page.

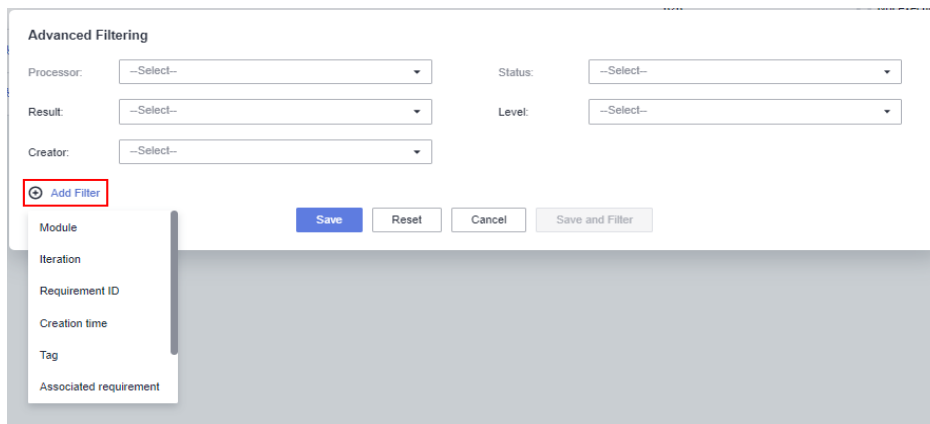


You can also click **Save and Filter**. In the dialog box that is displayed, enter the filter name and click **OK**. The saved filter is added to the **All cases** drop-down list.



**Step 5** (Optional) If advanced filter criteria still do not meet requirements, click **Add Filter**, select a filter item from the drop-down list box as required, and click **Save and Filter**.

The filtering items are displayed on the page. Repeat step **Step 4** to complete the filtering.




----End

## 5.13 Customizing Columns to Be Displayed in the Test Case List

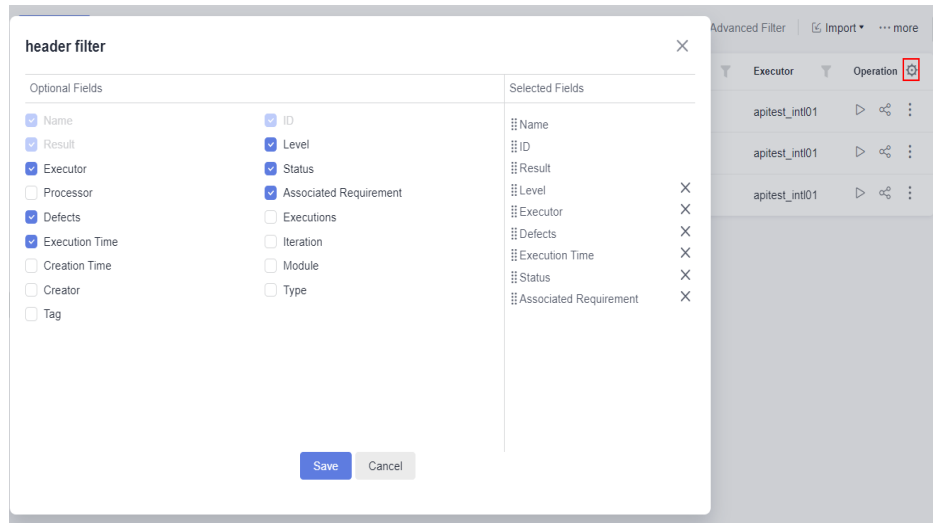
CodeArts TestPlan supports customizing columns to be displayed in the test case list. The following procedure uses manual test cases as an example.

**Step 1** 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 **Manual Test** tab page, click  in the last column of the test case list. In the dialog box that is displayed, you can configure the columns to be displayed or hidden, and drag the selected fields to rearrange the column display sequence.





----End

# 6 Testing Execution

---

- [6.1 Introduction](#)
- [6.2 Manual Test Suite](#)
- [6.3 API Automation Test Suite](#)
- [6.4 Automation Custom Test Suite](#)

## 6.1 Introduction

In the test execution stage, the test suite is executed to check whether the tested system meets the expected result, record the test result, and detect product problems and defects, helping R&D engineers analyze and locate problems.

A test suite is used to assign test cases to specified test executors.

The following sections introduce the details:

- [6.2 Manual Test Suite](#)
- [6.3 API Automation Test Suite](#)

## 6.2 Manual Test Suite


### 6.2.1 Creating a Manual Test Suite

#### Prerequisites

Manual test cases have been created.

#### 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 Execution**. By default, **Test case library** and **Baseline** are displayed.



- Step 3** Click  next to **Test case library** and select a test plan as required.
- Step 4** On the **Manual Test** tab page, click **Create Test Suite** in the upper left corner. The creation page is displayed.
- Step 5** Enter basic information such as the name, click **Add Case** or **Add now**, select the test case to be tested, click **OK**, and click **Save**.

**Create Manual Test Suite**

Name:

Description:

Test Cases: + Add Case

Case Name	ID	Processor	Result	Status 
 No cases in the test suite. <a href="#">Add now</a>				



----End

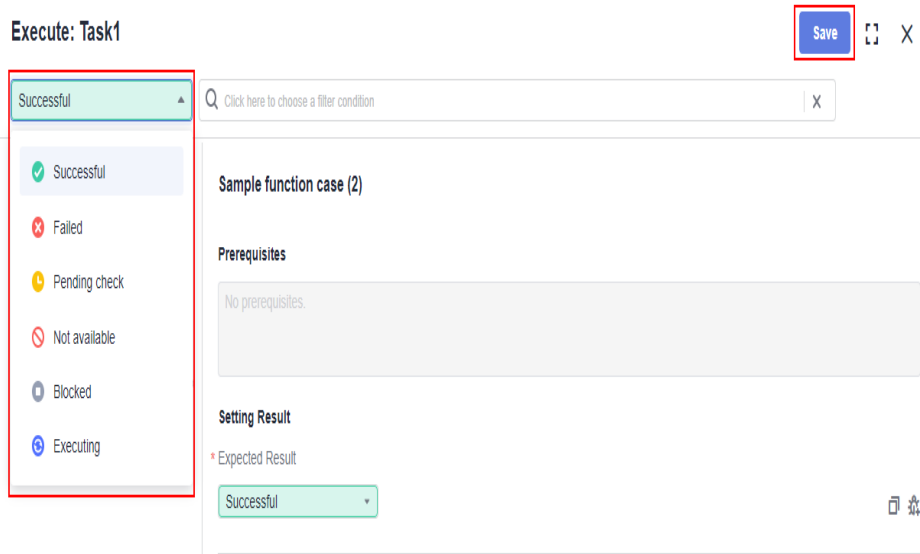
## 6.2.2 Executing a Manual Test Suite

### Prerequisites


A manual test suite has been created.

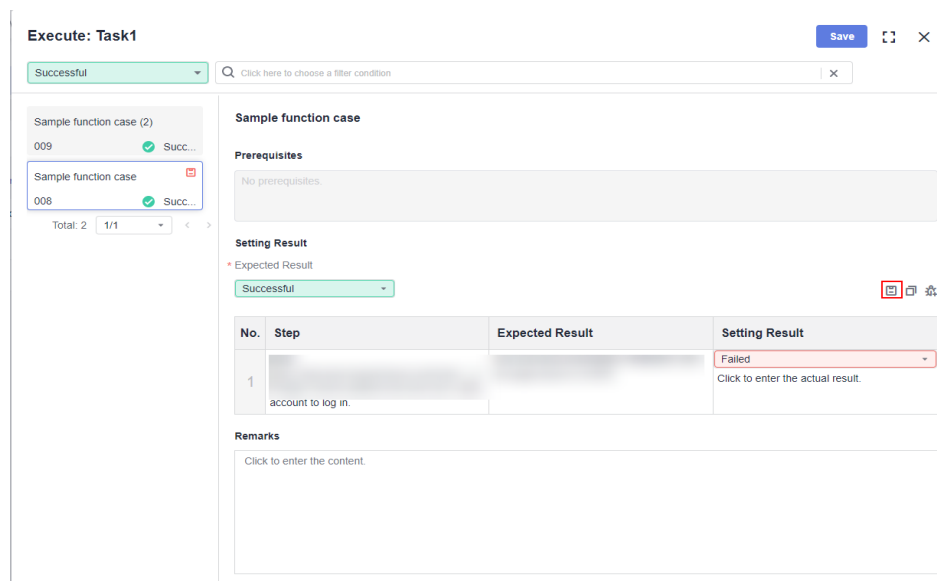
### Procedure

- 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 Execution**. **Test case library** is displayed by default.
- Step 3** Click  next to **Test case library** and select a test plan as required.
- 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. (You can switch cases in the suite and repeat this step until all cases in the manual test suite are executed.)
- Step 6** Set the manual test suite result and click **Save** in the upper right corner.



**NOTE**

Set the case result or switch the case to automatically save the result. If you modify the step result, description, or case execution remarks after saving the case result, click  to save the modification.



**Step 7** After the execution is complete, you can view the execution result in the **Result** column of the test case list.

----End


## 6.3 API Automation Test Suite

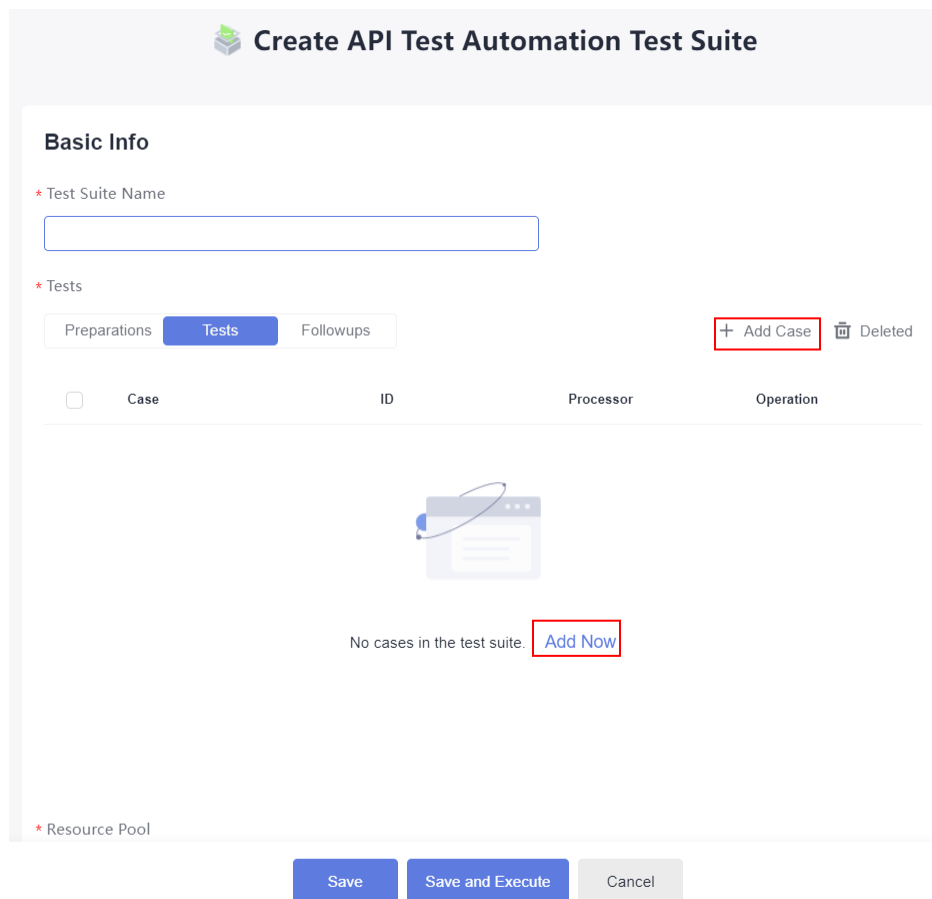
## 6.3.1 Creating an API Automation Test Suite

### Prerequisites

API automation test cases have been created.

### 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 Execution**.
- Step 3** Click  next to **Test case library** and select a test plan as required.
- Step 4** On the **Auto API Test** tab page, click **Create Test Suite** in the upper left corner. The creation page is displayed.
- Step 5** Enter basic information such as the name, click **Add Case** or **Add Now**, and select the test case to be executed.



**Create API Test Automation Test Suite**

**Basic Info**

\* Test Suite Name

\* Tests

Preparations Tests Followups

+ Add Case Deleted

Case	ID	Processor	Operation
No cases in the test suite. Add Now			

\* Resource Pool

Save Save and Execute Cancel

- Step 6** Complete the execution settings as required and click **Save**.

Configurati on Item	Manda tory	Description
Resource Pool	Yes	The resource pool allows you to access your own execution resources. When executing a task, you can select an agent in the resource pool to execute the task. This improves the task execution efficiency and does not depend on public resources of CodeArts.
ID	No	You can enter a number of 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 case. The module list comes from the project settings. For details, see Setting Modules.
Processor	No	Person who needs to complete the test task.
Environmen t Parameters	No	Environment parameters can be referenced by parameters, checkpoints, variables, and URLs of test steps in the entire project.
Max. Duration	Yes	The longest time taken for executing a test case. Cases exceeding this limit will fail. The task continues until the last case is complete. The value can be minute-level, hour-level, or day-level.
Configure Test Suite Parameters	No	Add new parameters or modify existing global parameters of the test suite.
Execute	Yes	There are two types. By default, the task is executed only once. <ul style="list-style-type: none"> <li>● <b>Once:</b> The test suite is executed only once.</li> <li>● <b>Regularly:</b> After you set an execution frequency, the test suite is executed periodically. Set this parameter to daily.</li> </ul>
Start	Yes	There are two execution modes. By default, the task is executed immediately. <ul style="list-style-type: none"> <li>● <b>Immediately:</b> The task is executed immediately after it is executed.</li> <li>● <b>At specific time:</b> The task is executed after a specified time.</li> </ul>
Sequence	Yes	There are two modes. The default mode is <b>Serial</b> . <ul style="list-style-type: none"> <li>● <b>Serial:</b> Test cases in the API test suite are executed in serial mode.</li> <li>● <b>Parallel:</b> Test cases in the API test suite are executed in parallel mode.</li> </ul>

----End

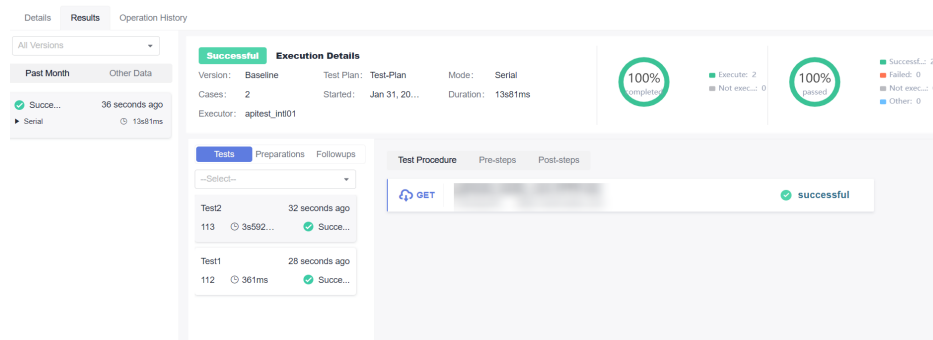
## 6.3.2 Executing an API Automation Test Suite

### Prerequisites

An API automation test suite has been created.

### 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 Execution**. **Test case library** is displayed by default.
- Step 3** Click **▼** next to **Test case library** and select a test plan as required.
- 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 **Result** column.
- Step 5** In the test suite list, click **🕒** in the **Operation** column. On the **Results** tab page that is displayed, you can view the execution details of historical test cases executed by the test suite.



----End

## 6.4 Automation Custom Test Suite

### 6.4.1 Creating a Custom Automation Suite

#### Prerequisites

Custom automation test cases have been created.

## Procedure

- 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 Execution**.
- Step 3** Click ▼ next to **Baseline** and select a version.
- Step 4** Click ▼ next to **Test case library** and select a test plan.
- Step 5** On the custom execution type tab page, click **Create Test Suite** at the upper left corner or **Create Test Suite** in the middle of the page. The creation page is displayed.
- Step 6** Enter basic information such as the name, click **Add Case** or **Add Now**, and select the test case to be executed.
- Step 7** Complete the execution settings, and click **Save**. The custom automation suite is created.

Configuration Item	Mandatory	Description
Resource Pool	Yes	You can add your own execution resources to the resource pool. When executing a task, you can select an agent in the pool to execute the task, improving efficiency without depending on the public execution resources of the software development platform.
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 to which a test case belongs. The module list comes from the project settings.
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.
Environment Parameters	No	Environment parameters can be referenced by parameters, checkpoints, variables, and URLs of test steps in the entire project.
Max. Duration	Yes	The longest time taken for executing a test case. Cases exceeding this limit will fail. The task continues until the last case is complete. The value can be set to minutes.
Configure Test Suite Parameters	No	Add new parameters or modify existing global parameters of the test suite.



Configuration Item	Mandatory	Description
Execution Policy	Yes	There are two types. By default, the task is executed only once. <ul style="list-style-type: none"><li>• <b>Once:</b> The test suite is executed only once.</li><li>• <b>Regularly:</b> After you set an execution frequency, the test suite is executed periodically. The value can be minute-level, hour-level, or day-level.</li></ul>
Start	Yes	There are two options. By default, the task is executed immediately. <ul style="list-style-type: none"><li>• <b>Immediately:</b> The task is executed immediately.</li><li>• <b>At specific time:</b> The task is executed at a specified time.</li></ul>
Sequence	Yes	There are two modes. The default mode is <b>Serial</b> . <ul style="list-style-type: none"><li>• <b>Serial:</b> API automation test cases or API test suite cases are distributed to one executor for serial execution.</li><li>• <b>Parallel:</b> API automation test cases or API test suite cases are distributed to multiple executors for parallel execution.</li></ul>

----End


## 6.4.2 Executing a Custom Automation Suite

### Prerequisites

Custom automation test suites have been created.

### Procedure

- 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 Execution**.
- Step 3** Click ▼ next to **Baseline** and select a version.
- Step 4** Click ▼ next to **Test case library** and select a test plan.
- Step 5** 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 **Result** column.

**Step 6** In the test suite list, click  in the **Operation** column. On the **Results** tab page that is displayed, you can view the historical execution details of test cases executed by the test suite.

----End

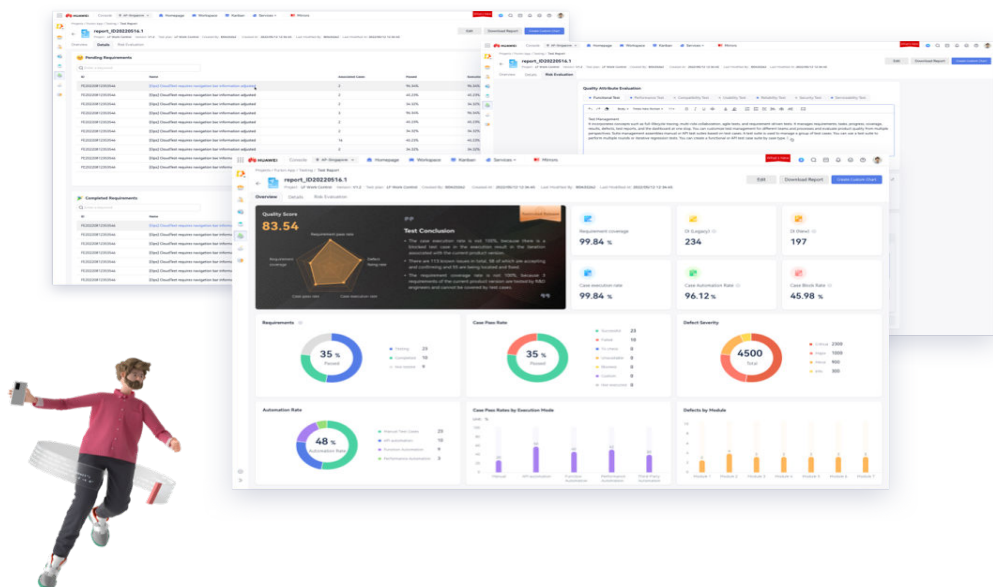
# 7 Quality Report and Assessment

- [7.1 Introduction](#)
- [7.2 Quality Report — Project Dashboard](#)
- [7.3 TestPlan Homepage - Personal Dashboard](#)
- [7.4 Custom Test Report](#)
- [7.5 Evaluating Test Quality](#)

## 7.1 Introduction

Management guru Peter Drucker once said "what gets measured gets managed". This principle applies to quality of all commercial products. You can use mature models and guidelines to evaluate product quality in a scientific, objective, visualized, and measurable manner, achieving continuous product improvement and high quality goals.

Huawei Cloud CodeArts TestPlan provides more than 10 quality metrics, such as the requirement coverage rate, requirement pass rate, case execution rate, and legacy DI, to evaluate functions, performance, and reliability. A test period can be shortened from days to hours.





## 7.2 Quality Report — Project Dashboard

The project dashboard displays the requirement coverage rate, number of defects, test case pass rate, and test case completion rate corresponding to the test case library and test plan, as well as detailed information such as defects associated with the test cases.

### Quality Report of the Test Case Library

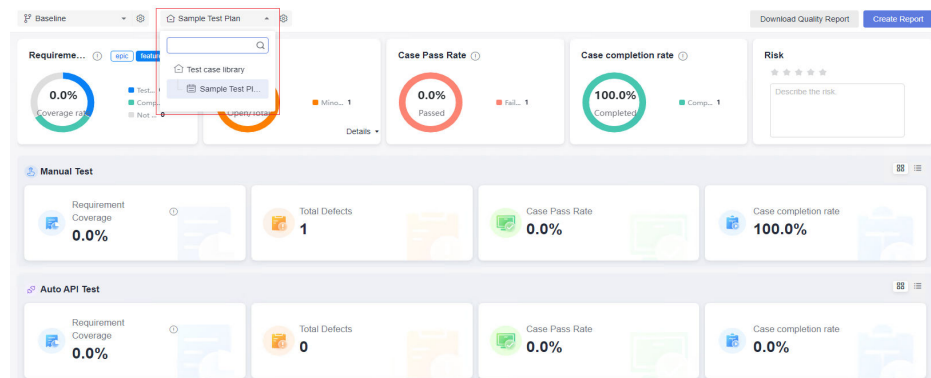
Log in to the CodeArts homepage and search for and access the target project. In the navigation pane, choose **Testing > Quality Report**. The **Quality Report** page is displayed by default.

Report Item	Description
Requirement Coverage	<p>Percentage of tested requirements in the selected iteration and module. This reflects function test coverage.</p> <ul style="list-style-type: none"><li>● <b>Not tested:</b> No test case is associated with the requirement, or each associated test case is not complete.</li><li>● <b>Testing:</b> Some test cases associated with the requirement are uncompleted.</li><li>● <b>Completed:</b> All test cases associated with the requirement are completed.</li><li>● <b>Requirement coverage = Completed/Total</b></li></ul> <p><b>NOTE</b> In Scrum projects, only statistics on the Feature and Story requirements are collected by default. To collect statistics on the Epic requirements, select <b>epic</b>.</p>
Defects	<p>Collects statistics on the number of unresolved defects and the total number of defects in the selected iteration and module. The statistics are grouped by defect severity.</p> <p><b>NOTE</b> The number of unresolved defects includes the following:</p> <ul style="list-style-type: none"><li>● Number of defects in the <b>To do</b> and <b>Doing</b> states in a Scrum project.</li></ul>
Case Pass Rate	<p>Pass rate of test cases in the selected iteration and module. The test case pass rate and defects reflect the overall product quality. The statistics are grouped based on the execution result. The test cases that are not executed are included in the <b>Not executed</b> group.</p> <p><b>Case pass rate = Number of successful test cases/Total number of test cases</b></p>
Manual Test	<p>Collects statistics on the requirement coverage rate, total number of defects, case pass rate, and case completion rate associated with manual test cases in the selected iteration and module.</p>
Auto API Test	<p>Collects statistics on the requirement coverage rate, total number of defects, case pass rate, and case completion rate associated with API automation test cases in the selected iteration and module.</p>

Report Item	Description
Defect List	List of defects associated with test cases in the selected iteration and module. Click  in the upper right corner of the Manual Test or Auto API Test area to view the list. You can filter defects by defect name and ID. You can also click a defect name to go to the defect details page.
Test Case List	List of test cases in the selected iteration and module. Click  in the upper right corner of the Manual Test or Auto API Test area to view the list. You can filter test cases by test case name and ID. You can also click a test case name to go to the case details page.

## 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 test plan quality report page is displayed.



Compared with the quality report of the test case library, the quality report of the test plan is not filtered by iteration or module. The statistical report of the test case completion rate and the risk description of the test plan are added. Other report items are the same as those of [Quality Report of the Test Case Library](#).

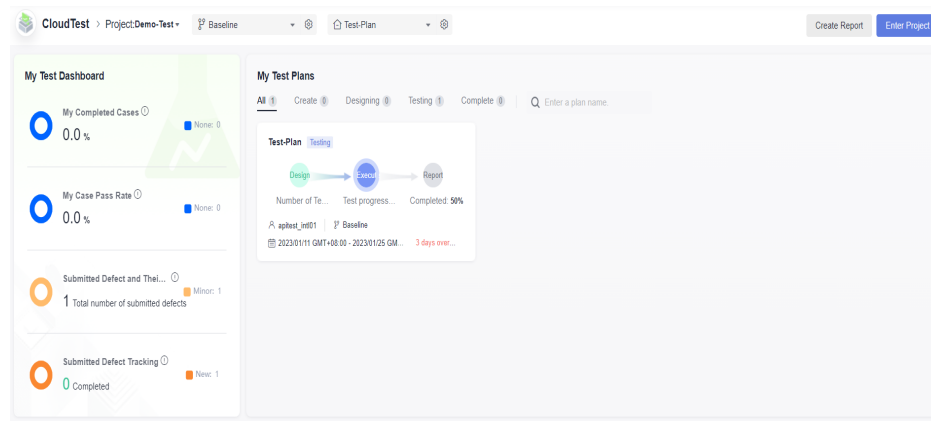
Report Item	Description
Case Completion Rate	Number of completed test cases in the selected plan. This reflects the test progress. The statistics are grouped by case status.  Completion rate = Number of completed cases/Total number of cases
Risk	Risk of the test plan. You can evaluate the risk level of the test plan and add risk description.
Manual Test	Requirement coverage rate, total number of defects, and pass rate of manual test cases.

Report Item	Description
Auto API Test	Total number of defects, pass rate, and completion rate of API automation test cases.

## 7.3 TestPlan Homepage - Personal Dashboard

Log in to the CodeArts homepage. On the top navigation bar, choose **Services > TestPlan**. The CodeArts TestPlan page displays the personal dashboard. It contains statistics that are processed by the current login user, including:

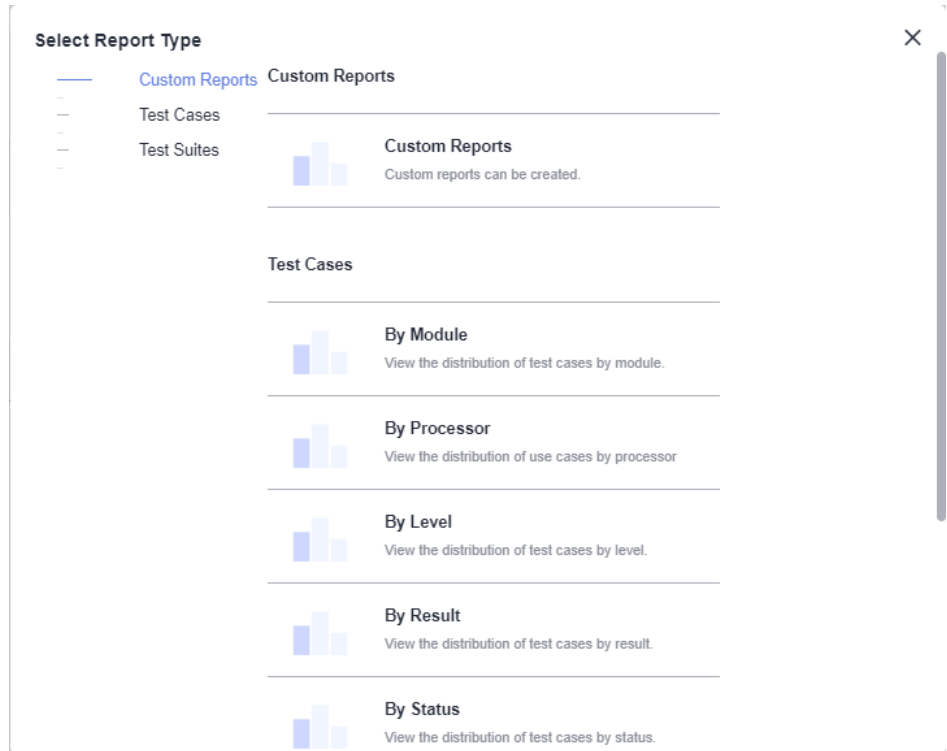
- Personal case completion rate, case pass rate, submitted defect status, submitted defect severity, defect statistics based on the test case library and test plan in the project
- Personal test plans, test tasks, and submitted defects based on the test case library and test plan in the project



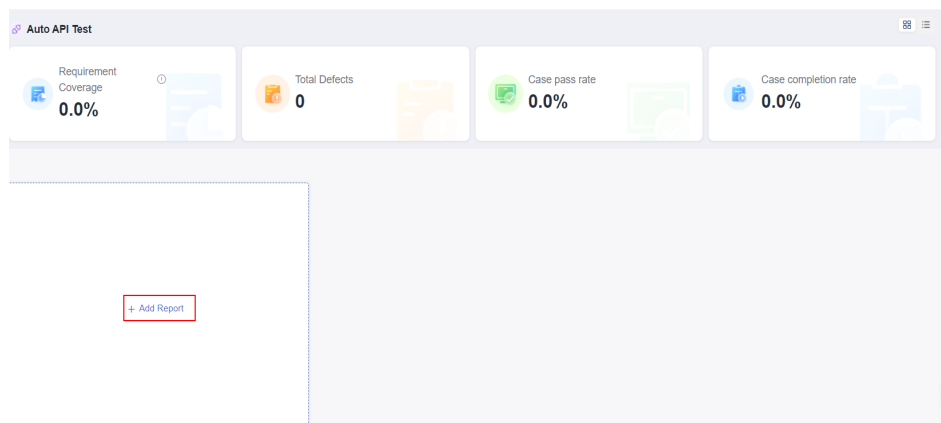
## 7.4 Custom Test Report

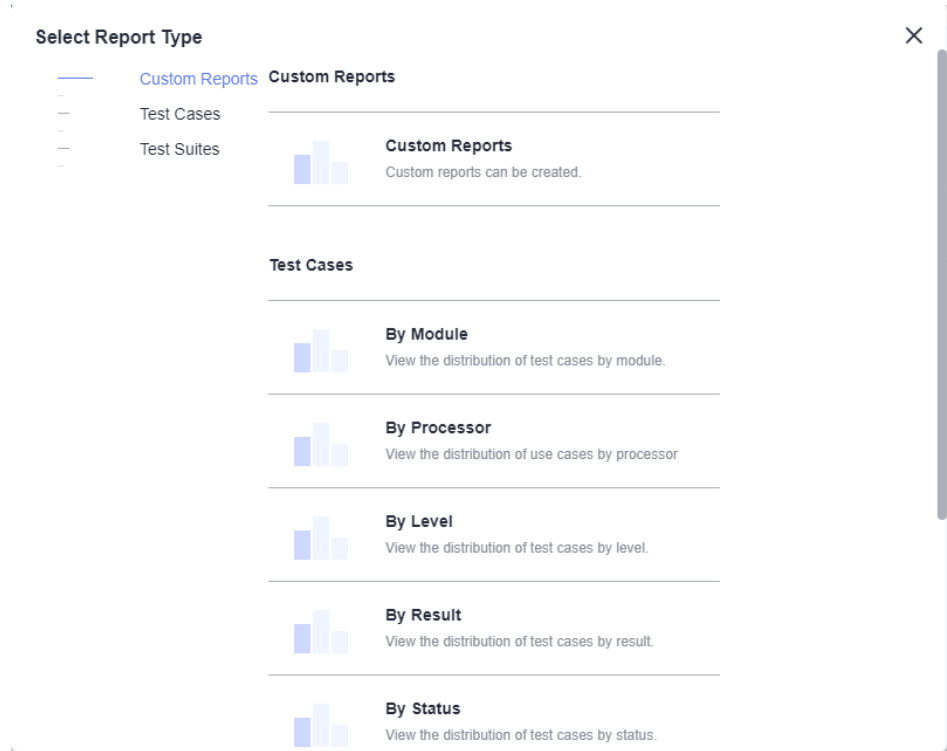
CodeArts TestPlan allows you to create a custom test report in either of the following ways:

- 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 custom test report of the current project in the lower left corner.
  - c. On the **Select Report Type** page that is displayed, select the type of the report to be created as required.



- 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 the test case library or a test plan, click **Add Report** in the lower left corner of the page, and select the type of the report to be created as required.

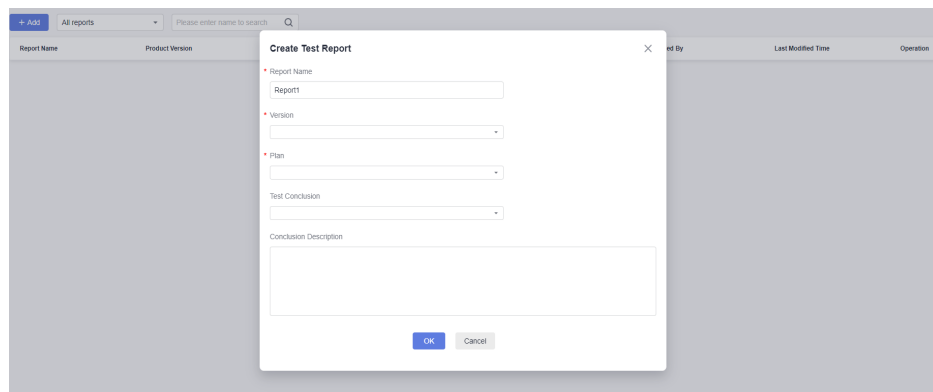




## 7.5 Evaluating Test Quality

Log in to the CodeArts homepage and search for and access the target project. In the navigation pane, choose **Testing > Quality Assessment**.

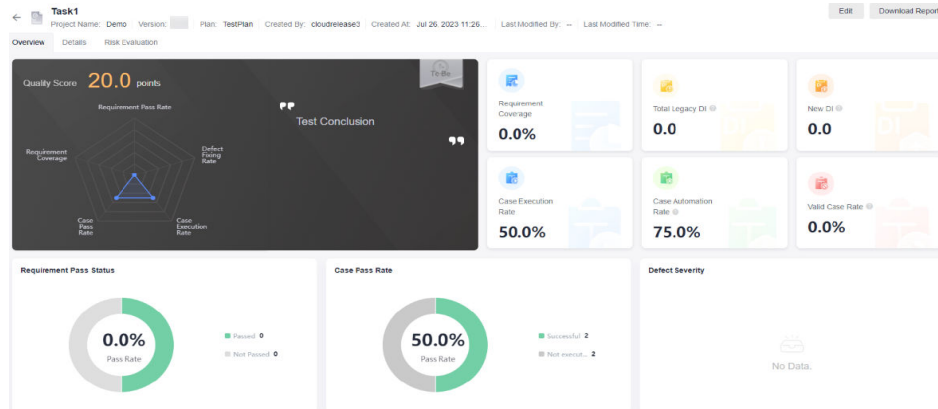
Click **Add** in the upper left corner of the page. Enter the report name, select the version, test plan, and test conclusion, and click **OK**.



### 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.





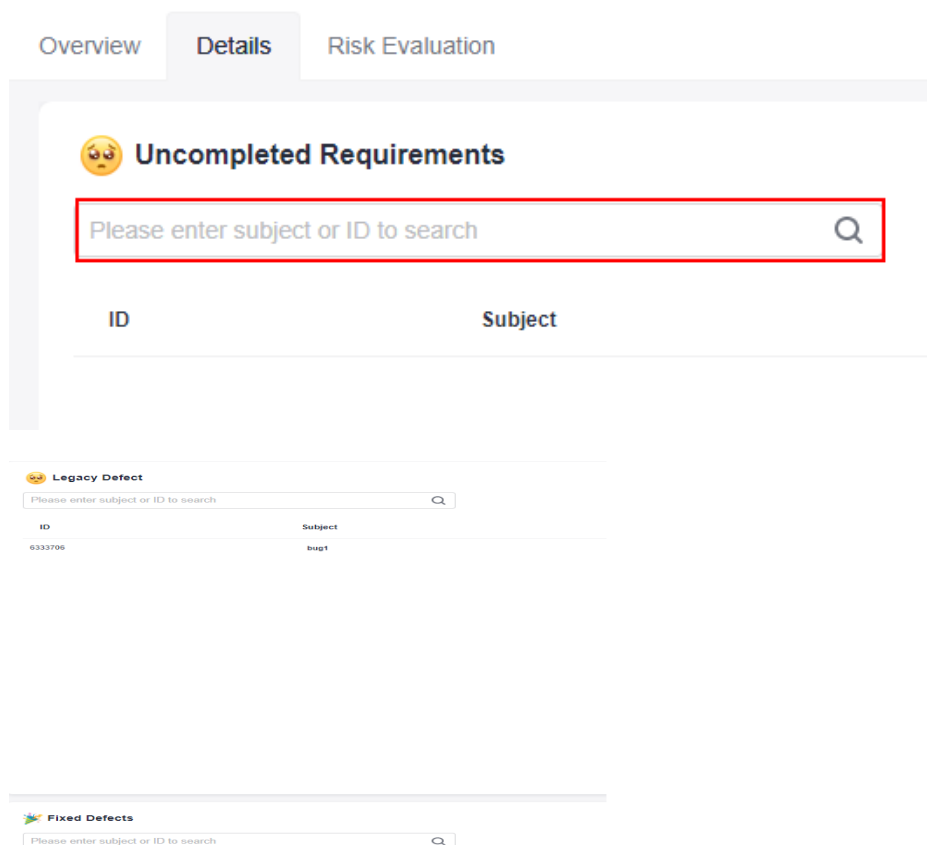
Report Item	Description
Requirement Coverage	The requirement coverage rate reflects the test coverage of function points. The test coverage rate of all requirements associated with the selected plan is calculated. Requirement coverage rate = Number of completed requirements/Total number of requirements
Total Legacy DI	DI value calculated based on all known defects in this version. <b>NOTE</b> 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	DI value calculated based on the outstanding defects associated with the test plan.
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 non-manual test cases to all test cases. Case automation rate = (Total number of test cases in the test plan – Number of manual test cases in the test plan)/Total number of test cases in the test plan
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	A requirement is passed if the status of all test cases associated with the requirement is <b>Successful</b> . Requirement pass rate = Number of passed requirements/Total number of requirements
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.

Report Item	Description
Automation Rate	Calculates the proportion of non-manual test cases to all test cases in the test plan.
Case Pass Rate by Type	Displays the pass rate of test cases of different test types.
Defects by Module	The number of defects is displayed by module.

## Details

On the **Details** page, you can view the outstanding and completion status of requirements and defects in the test plan.

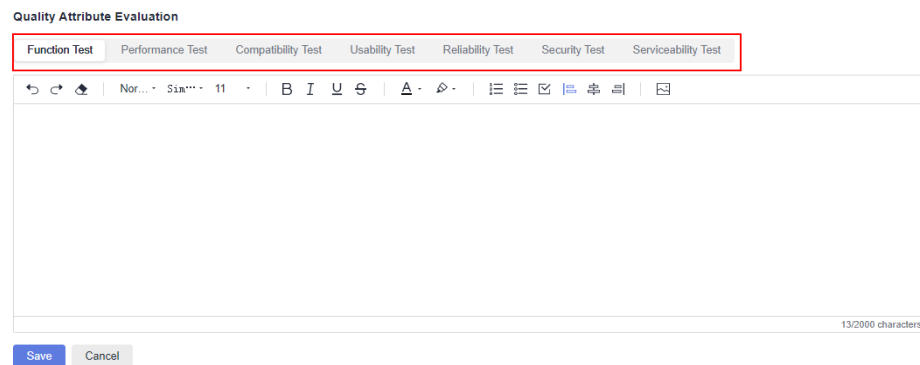
Enter a title or number in the search box to search for the corresponding requirement or defect.



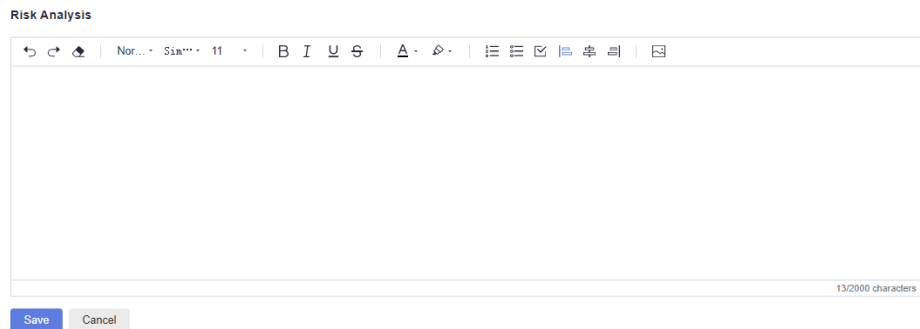
## Evaluating Risks

On the **Risk Evaluation** page, you can configure **Quality Attribute Evaluation** and **Risk Analysis**, or customize modules to enter related report information.

- **Quality Attribute Evaluation**  
Select a test type on the top, click the text box area, enter evaluation information, and click **Save**.

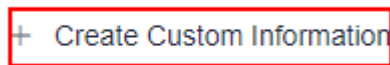




- **Evaluating Risks**  
You can evaluate risks based on the test progress. Click the text box, enter risk analysis information, and click **Save**.

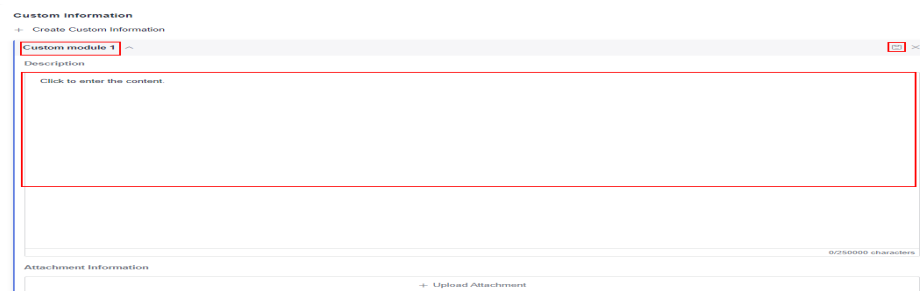


- **Custom information**  
Click **Create Custom Information** and add an information module.

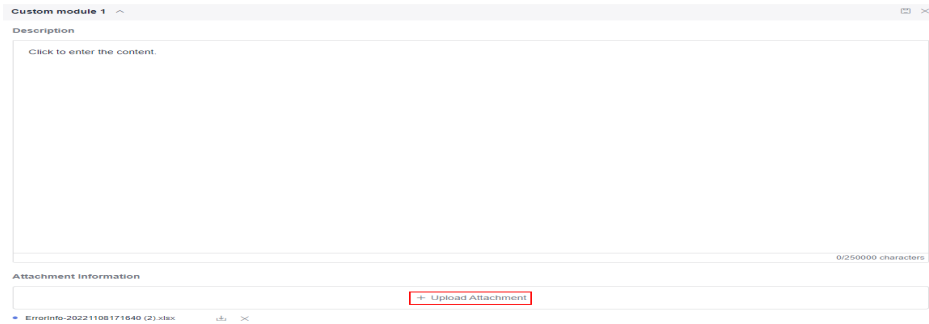
## Custom Information



You can click the module name to redefine the module title, enter the related description, and click  to save the modification. You can click  to delete the new module.



Click **Upload Attachment** to upload a local file to the customized module.

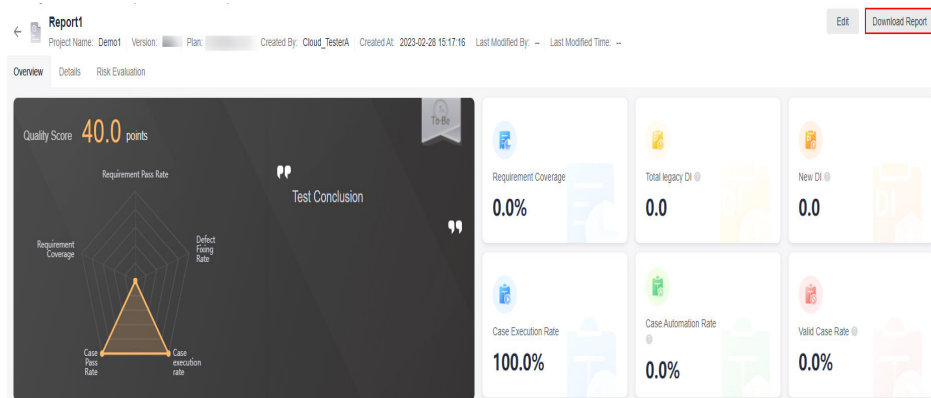


**NOTE**

- 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

Click **Download Report** in the upper right corner of the page to download the created test evaluation report in PDF format to the local PC.



# 8 Settings

- [8.1 Notification](#)
- [8.2 User Management](#)
- [8.3 Test Case Settings](#)
- [8.4 Test Suite Settings](#)
- [8.5 Setting a Custom Execution Type](#)
- [8.6 Other Settings](#)

## 8.1 Notification

On CodeArts TestPlan, you can configure whether to send notifications for each operation.

- 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 **Notification** tab page, configure whether to send service notifications and emails for system events.

Event type	Pop-up Notification	Email	Recipient
Assign Case Handler	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Delete Directory	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Delete Case	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Update Case	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Create Test Suite	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Update Test Suite	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Delete Test Suite	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Create Report	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Update Report	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Delete Report	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Edit Risk Info	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Create Plan	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Change Plan	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Delete Plan	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--
Execute Manual Test Case	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	--

**Notification sending policy**

<b>Event Type</b>	<b>Operator be Notified</b>	<b>Recipient</b>
Assign Case Handler	No	Test case handler
Update Case	No	Project manager, test manager, test case creator, and test case handler
Delete Case	No	Project manager, test manager, project creator (if there is no project manager or test manager), and test case handler
Delete Directory	No	Project manager, test manager, and project creator (if there is no project manager or test manager)
Create Test Suite	No	Test suite handler
Update Test Suite	No	Test suite creator and handler
Delete Test Suite	No	Project manager, test manager, project creator (if there is no project manager or test manager), and test suite handler
Create Report	No	Project manager, test manager, and project creator (if there is no project manager or test manager)
Update Report	No	Project manager, test manager, project creator (if there is no project manager or test manager), and custom report creator
Delete Report	No	Project manager, test manager, project creator (if there is no project manager or test manager), and custom report creator
Edit Risk Info	No	Project manager, test manager, project creator (if there is no project manager or test manager), test plan creator, and test plan handler
Create Plan	No	Project manager, test manager, project creator (if there is no project manager or test manager), and test plan handler

Event Type	Operator be Notified	Recipient
Change Plan	No	Project manager, test manager, project creator (if there is no project manager or test manager), test plan creator, and test plan handler
Delete Plan	No	Project manager, test manager, project creator (if there is no project manager or test manager), test plan creator, and test plan handler
Execute Manual Test Case	No	Test case creator and handler
Send Quality Report upon Plan Completion	Yes	Custom notification recipient on the CodeArts TestPlan settings page (otherwise, project manager and test manager)
Comment Notification	No	User who has been tagged for his or her comments on the test case details
API Test Cases Completed	Yes	API automation test case executor
API Test Suite Completed	Yes	API automation test suite executor

----End

## 8.2 User Management

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

## 8.3 Test Case Settings

If the preset test case 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 Case Settings** tab page, click **Add Status** or **Add Result** and configure the new case status and result as required.

----End

## 8.4 Test Suite Settings

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

## 8.5 Setting a Custom Execution Type

The execution type is used to manage the repository URL and version of the automation WebUI test.

### Prerequisites

A repository has been created in CodeArts Repo.

### Procedure

- 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 **Settings > Testing**. Click the **Execution Type Setting** tab.
- Step 3** Click **Add Execution Mode** in the upper left corner. In the dialog box that is displayed, set the repository URL and branch name.

----End



## 8.6 Other Settings

After DNS mapping is configured, if the URL request path in the API automation 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 service 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 **Other** tab and select the agent pool required for API test case execution (**DEFAULT** is the default public agent pool).

**Step 4** Set **Request Timeout Interval**. The default value is **10000** ms.

**Step 5** 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 (.)

----End

# 9 Roles and Operation Permissions

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

<b>Operation/Role</b>	Project creator	Project manager	Test manager	Developer	Tester	Participant	Viewer	Operation manager
<b>Version Management</b>	Create, modify, and delete a version.			View the version.				
<b>Test Plan Editing</b>	Create, delete, edit, and view test plans; add cases to and remove cases from test plans.			Create a test plan, delete a test plan created by yourself, edit a test plan, view a test plan, and add cases to a test plan.		View test plans.		
<b>Associated Work Items of Test Plans</b>	Associate defects, update associated defects, disassociate defects, query defects associated with test plans, add associated requirements, remove associated requirements, and view requirements associated with test plans.					Query defects associated with test plans and view requirements associated with test plans.		

<b>Test Cases</b>	Import test cases; merge test cases into the baseline test case library; import, create, delete, edit, view, execute, and stop test cases; create, delete, modify, view test case filters; restore deleted test cases.	Import test cases; create, delete your own, edit, view, execute, and stop test cases; create, delete, modify, and view test case filters; restore deleted test cases.	View test cases and use test case filters.
<b>Test Suites</b>	Create, delete, edit, view, execute, and stop test suites, and specify handlers.	Create, edit, view, execute, and stop test suites, delete personal test suites, and specify handlers.	View test suites.
<b>Test Case Attachment Management</b>	Associate and disassociate documents, and upload and download attachments.		Download attachments.
<b>Keyword</b>	Import, view, delete, and reference keywords.		View keyword details
<b>Common Parameters</b>	Create, delete, modify, and view common parameters.		View parameters.
<b>Encryption Parameters</b>	Create, delete, modify, and view encryption parameters.	Create encryption parameters; modify, delete, and view personal encryption parameters; modify others' encryption parameter names.	-

<b>Quality Reports</b>	View, export, download, create, update, and delete quality reports.		View, export, download, create, and update quality reports; delete personal quality reports.	View quality reports.
<b>Settings</b>	Set message notifications, set function switches, manage users, customize test cases, and customize test suites.	Set message notifications, set function switches, customize test cases, and customize test suites.	Set function switches.	View settings.