CodeArts PerfTest

User Guide

Issue 01

Date 2026-02-12

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Huawei Cloud Computing Technologies Co., Ltd.

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

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

Contents

1 CodeArts PerfTest Use Process

CodeArts PerfTest provides performance test services for cloud applications that are built based on HTTP, HTTPS, TCP, UDP, HLS, RTMP, WebSocket, MQTT, or HTTP-FLV. CodeArts PerfTest quickly simulates service peaks with a large number of concurrent users. It allows you to define the contents and time sequences of packets and supports complex combinations of multiple transactions. After tests are complete, CodeArts PerfTest provides professional test reports to evaluate your service quality.

Basic Operation Process

Figure 1-1 Basic operation process

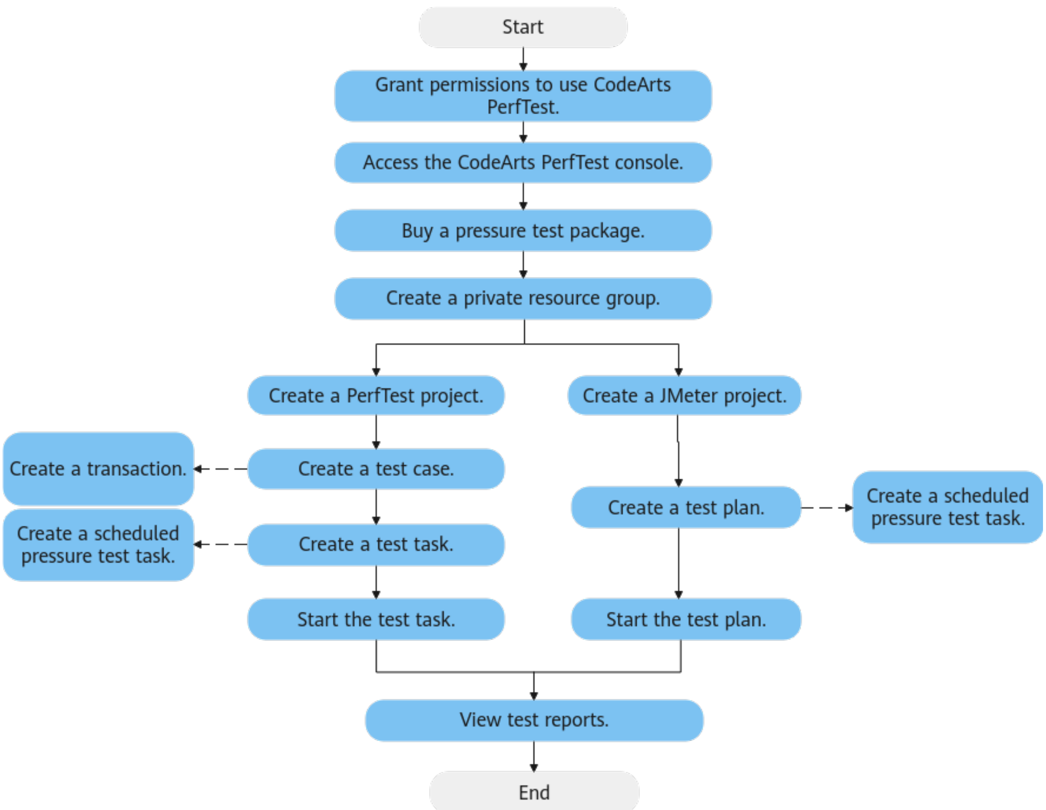


Table 1-1 CodeArts PerfTest operation process description

Process	Description
Permissions Management via IAM	This section describes how to authorize IAM users to use CodeArts PerfTest.
Accessing the CodeArts PerfTest Dashboard Page	This section describes how to access the CodeArts PerfTest dashboard page.
Buying a CodeArts PerfTest Package	This section describes how to buy a CodeArts PerfTest package. **NOTE** The CN-Hong Kong region supports only pay-per-use billing. No pressure test package purchase is required.
Test Resource Group Management	This section describes how to create and manage private resource groups.
PerfTest Project Management	CodeArts PerfTest test projects are classified into PerfTest projects and JMeter projects. This section describes how to create a PerfTest project, create a test case and a test task in the PerfTest project, start the test task, and view the test report.
JMeter Test Project Management	This section describes how to create a JMeter project, create a test plan, start the test plan, and view the test report.

2 Permissions Management via IAM

2.1 Using IAM Roles to Grant Access to CodeArts PerfTest

System-defined permissions in role/policy-based authorization provided by **IAM** let you control access to CodeArts PerfTest. With IAM, you can:

- Create IAM users for personnel based on your enterprise's organizational structure. Each IAM user has their own identity credentials for accessing CodeArts PerfTest resources.
- Grant users only the permissions required to perform a given task based on their job responsibilities.
- Entrust a Huawei Cloud account or a cloud service to perform efficient O&M on your CodeArts PerfTest resources.

If your Huawei Cloud account meets your permissions requirements, you can skip this section.

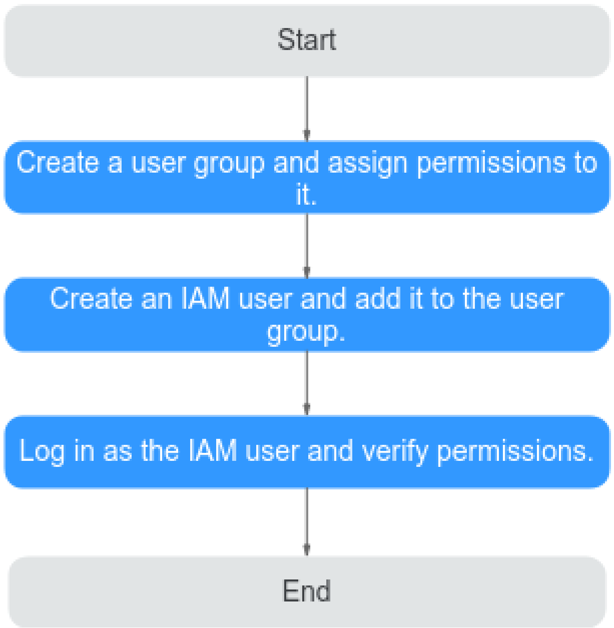
Figure 2-1 shows the process flow of role/policy-based authorization.

Prerequisites

Before granting permissions to user groups, learn about system-defined permissions in **Role/Policy-based Permissions Management** for CodeArts PerfTest. To grant permissions for other services, learn about all **system-defined permissions** supported by IAM.

Process Flow

Figure 2-1 Process of granting CodeArts PerfTest permissions using role/policy-based authorization



1. On the IAM console, **create a user group and grant it permissions**.

   Create a user group on the IAM console, and assign the **CodeArts PerfTest Operator** policy to the group.
2. **Create an IAM user and add it to the created user group**.

   On the IAM console, create a user and add it to the user group created in **1**.
3. **Log in as the IAM user** and verify permissions.

   In the authorized region, perform the following operations:

   In the service list, choose **CodeArts PerfTest**. The **Dashboard** page of the CodeArts PerfTest console is displayed by default. Choose **PerfTest Projects** in the navigation pane and click **Create Test Project** on the right. If a message appears indicating that you have insufficient permissions to create a test project, the **CodeArts PerfTest Operator** policy is in effect.

2.2 Using IAM Identity Policies to Grant Access to CodeArts PerfTest

System-defined permissions in identity policy-based authorization provided by **IAM** let you control access to CodeArts PerfTest. With IAM, you can:

- Create IAM users or user groups for personnel based on your enterprise's organizational structure. Each IAM user has their own identity credentials for accessing CodeArts PerfTest resources.
- Grant users only the permissions required to perform a given task based on their job responsibilities.

- Entrust a Huawei Cloud account or a cloud service to perform efficient O&M on your CodeArts PerfTest resources.

If your Huawei Cloud account meets your permissions requirements, you can skip this section.

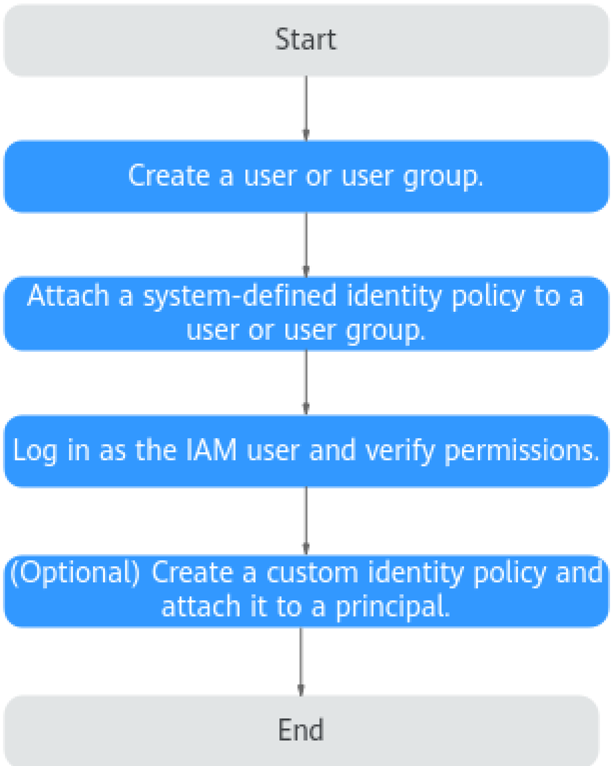
Figure 2-2 shows the process flow of identity policy-based authorization.

Prerequisites

Before granting permissions, learn about system-defined permissions in **Identity Policy-based Authorization** for CodeArts PerfTest. To grant permissions for other services, learn about all **system-defined permissions** supported by IAM.

Process Flow

Figure 2-2 Process of granting CodeArts PerfTest permissions using identity policy-based authorization



1. On the IAM console, **create an IAM user** or **create a user group**.

   On the IAM console, create a user or user group.

2. **Attach a system-defined policy to the user or user group**.

   Assign the permissions defined in the system-defined identity policy **CodeArtsPerfTestReadOnlyPolicy** to the user or group, or attach the system-defined identity policy to it.

3. **Log in as the IAM user** and verify permissions.

   In the authorized region, perform the following operations:

In the service list, choose **CodeArts PerfTest**. The **Dashboard** page of the CodeArts PerfTest console is displayed by default. Choose **PerfTest Projects** in the navigation pane and click **Create Test Project** on the right. If a message appears indicating that you have insufficient permissions to create a test project, the **CodeArtsPerfTestReadOnlyPolicy** policy is in effect.

Example Custom Identity Policies

You can create custom identity policies to supplement the system-defined identity policies of CodeArts PerfTest.

To create a custom identity policy, choose either visual editor or JSON.

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy grammar.
- JSON: Create a JSON policy or edit an existing one.

For details, see **Creating a Custom Identity Policy and Attaching It to a Principal**.

The following lists examples of common CodeArts PerfTest custom identity policies.

- Example 1: Grant permission to add a PerfTest project on the console.

```
{
  "Version": "5.0",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeartsperftest:cpts:listPerfTestProject",
        "codeartsperftest:cpts:createPerfTestProject"
      ]
    }
  ]
}
```

- Example 2: Grant permission to deny a user from modifying PerfTest projects, cases, and directories.

  An identity policy with only "Deny" permissions must be used together with other identity policies. If the permissions granted to an IAM user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

  Assume that you want to grant the permissions of the **CodeArtsPerfTestFullAccessPolicy** identity policy to a user but want to prevent them from modifying PerfTest projects, cases, and directories. You can create a custom identity policy for denying the modification of PerfTest projects, cases, and directories, and attach this policy together with the **CodeArtsPerfTestFullAccessPolicy** policy to the user. As an explicit deny in any policy overrides any allows, the user can perform all operations on CodeArts PerfTest excepting modifying PerfTest projects, cases, and directories. Example identity policy denying the modification of PerfTest projects, cases, and directories:

```
{
  "Version": "5.0",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
```

```
        "codeartsperftest:cpts:updatePerfTestProject"
      ]
    }
  ]
}
```

- Example 3: Create a custom policy containing multiple actions.

  A custom identity policy can contain the actions of one or multiple services that are of the same type (global or project-level). Example policy containing multiple actions:

```
{
  "Version": "5.0",
  "Statement": [
    {
      "Action": [
        "codeartsperftest:cpts:listPerfTestProject",
        "cce:cluster:list"
      ],
      "Effect": "Allow"
    }
  ]
}
```

- Example 4: Grant permission to create a private resource group on the console.

```
{
  "Version": "5.0",
  "Statement": [
    {
      "Action": [
        "codeartsperftest:*:*",
        "aom:cmdbResources:list",
        "aom:metric:list",
        "aom:inventory:list",
        "apm:application:list",
        "apm:application:getMetric",
        "apm:application:getEnv",
        "apm:application:getInstance",
        "apm:application:getService",
        "apm:application:get",
        "cce:cluster:list",
        "cce:cluster:getCluster",
        "cce:release:create",
        "cce:release:delete",
        "cce:release:get",
        "cce:release:list",
        "cce:node:list",
        "cce:node:getNode",
        "ecs:cloudServers:showFlavor",
        "ecs:cloudServerFlavors:get",
        "ecs:flavors:get",
        "vpcep:endpoints:create",
        "vpcep:endpoints:delete",
        "vpcep:endpoints:list",
        "vpcep:endpoints:get"
      ],
      "Effect": "Allow"
    }
  ]
}
```

3 Accessing the CodeArts PerfTest Dashboard Page

Constraints

To use CodeArts PerfTest for a pressure test on a public website, you must be on the whitelist of this website. Otherwise, you must bear any and all consequences and legal liabilities.

Prerequisites

1. You have **signed up for a Huawei ID and enabled Huawei Cloud**.
2. Your account has permission to use CodeArts PerfTest. For details, see **Permissions Management via IAM**.

Logging In to the CodeArts PerfTest Console

Step 1 Log in to the **Huawei Cloud console**.

Step 2 Click and select a region.

Step 3 Click the service list in the upper left corner, search for **CodeArts PerfTest**, and click the corresponding result.

- If this is your first login, select the disclaimer and click **Apply now**. The CodeArts PerfTest console will be displayed.
- If this is not your first login, the CodeArts PerfTest console is displayed directly.

----End

4 Buying a CodeArts PerfTest Package

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the upper right corner of the **Dashboard** page, click **Buy Package**.

> **NOTE**
>
> The CN-Hong Kong region supports only pay-per-use billing. It neither supports package billing nor provides a package purchase page.

Step 3 Configure **Peak Concurrency**, **Duration**, and **Auto-renew** as required.

Step 4 Click **Next**.

Step 5 Confirm and pay for the order.

----End

5 Test Resource Group Management

5.1 CodeArts PerfTest Resource Group Configuration Suggestions

Test Resource Groups and Their Constraints

- Test resource groups are classified into shared resource groups and private resource groups. Shared resource groups are provided by the system by default, and private resource groups need to be created. To obtain the network segment of a shared resource group, submit a service ticket to after-sales.
- Execution nodes of the shared resource group have been bound with an elastic IP address (EIP). When the tested application has network access restrictions, use a private resource group.
- A shared resource group supports a maximum of 1,000 concurrent users and 100 Mbit/s bandwidth. If higher concurrency or bandwidth is required, use a private resource group.
- JMeter test tasks can use only private resource groups.

Suggestions on Using Nodes

- If an application is deployed on a node in a cluster, the node cannot be selected to create a private resource group. Do not run any applications or perform other functions on nodes used for test resource groups. Otherwise, applications may run abnormally.
- If you want to perform pressure tests on external services, bind an EIP to each execution node. If you want to debug external services, bind EIPs to both the debugging node and execution node. The test bandwidth is limited by the EIPs' bandwidth.
- Create at least two empty nodes. One is the execution node/executor (a target machine that a pressure test will be performed on and provides performance data during testing). The other serves as a debugging node to debug the execution node. Create nodes of the required specifications based on the number of concurrent users for a pressure test. For details about the recommended node specifications, see **Table 5-1** and **Table 5-2**. These

specifications are for reference only. Resource specification requirements for a pressure test are influenced by think time, protocol type, the size and number of requests and responses, response time, and result verification. Adjust the specifications as needed.

- In a PerfTest test project, one execution node with 8 vCPUs and 16 GB memory supports 10,000 concurrent users. In a JMeter test project, one execution node with 8 vCPUs and 16 GB memory supports 2,000 concurrent users.

Table 5-1 Recommended node specifications for PerfTest projects

Concurrent Users	Specifications	Quantity
0–5,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 4 vCPUs | 8 GB	1
5,001–10,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	1
10,001–20,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	2
20,001–30,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	3
30,001–40,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	4
40,001–50,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	5
More than 50,001	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	*n*

Table 5-2 Recommended node specifications for JMeter projects

Concurrent Users	Specifications	Quantity
0–1,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 4 vCPUs | 8 GB	1
1,001–2,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	1
2,001–4,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	2
4,001–6,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	3
6,001–8,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	4
8,001–10,000	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	5
More than 10,001	Debugging node: 4 vCPUs | 8 GB	1
	Execution node: 8 vCPUs | 16 GB	*n*

5.2 Creating a Private Resource Group

Creating a Private Resource Group

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **Resource Groups**.

Step 3 Click **Create Private Resource Group**.

Step 4 (Optional) If this is the first time you create a private resource group, grant CodeArts PerfTest permissions necessary for creating private resource groups.

Step 5 On the page for creating a resource group, perform the following operations based on the actual situation:

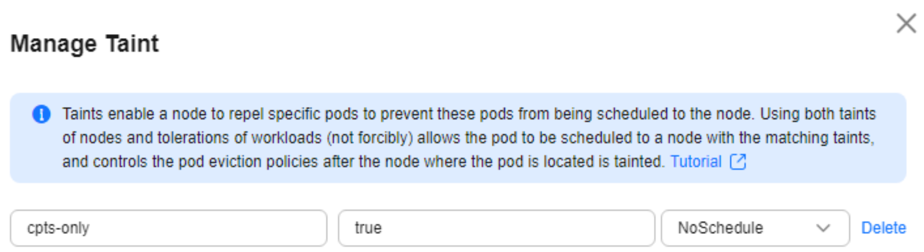
- If you do not have a CCE cluster, **create a cluster** and then **create a resource group**.
- If a CCE cluster is available, **create a resource group**.

Step 6 Create a cluster.

1. In the upper part of the page, click **create one** to access the page for buying a CCE cluster. For details about how to create a cluster, see **Buying a CCE Standard/Turbo Cluster**.

   Configure the cluster parameters as follows:

   – You are advised to use an independent CCE cluster for pressure tests to avoid configuration conflicts with CCE clusters in test or production environments. When you conduct a pressure test on a CCE cluster in a test or production environment, service loads may be scheduled to nodes that function as executors. To prevent this, go to the cluster's **Nodes** page, click the **Nodes** tab, click **More** > **Manage Taint** in the **Operation** column of the execution node, and set the node as a taint, as shown below.

   – The cluster management scale is related to the number of execution nodes. Create nodes of the corresponding specifications based on the number of concurrent users for pressure testing. For example, if 20 execution nodes are required, you can set the cluster scale to 50 nodes.

   – You are advised to select **Tunnel network** as the network model. **Container CIDR Block** and **Service CIDR Block** must be the same as those of the tested object.

   – CentOS is prone to IPVS- or Conntrack-related stability issues on a heavy-load network. Therefore, do not use CentOS as the operating system of cluster nodes when selecting IPVS. When both **IPVS** and **CentOS** are selected, network connection reuse may time out.

2. Click **Next: Select Add-on**. On the page displayed, select the add-ons to be installed during cluster creation. When selecting add-ons, retain the default settings for the test executor. For example, deselect unnecessary add-ons, such as NodeLocal DNSCache and Cloud Native Cluster Monitoring, to prevent the add-ons from occupying executor resources.

3. Click **Next: Configure Add-on**. On the displayed page, retain the default settings.

4. Click **Next: Confirm Settings**. Review the cluster configurations, select the disclaimer, and click **Submit**. It takes about 6 to 10 minutes to create a cluster.
5. After the cluster is created, return to the cluster management page and click **Create Node**. Set node parameters. For details about how to create a node, see **Creating a Node**.

   Configure the node parameters as follows:
   - A node must have at least 4 vCPUs and 8 GB memory.
   - The OS must be EulerOS.
   - At least two nodes (one debugging node and one execution node) are required. The number of nodes depends on the specifications of the pressure test object. For example, a pressure test for 100,000 concurrent users requires 21 nodes (one debugging node and 20 execution nodes), each with a minimum of 4 vCPUs and 8 GB of memory.
   - If the CCE cluster node and the tested application are not in the same VPC network, bind an EIP to the CCE cluster node. You can use existing EIPs. If no EIP is available, click **Auto assign** to create one. When EIPs are automatically created, you are advised to pay by traffic and set the bandwidth to a large value to avoid affecting the pressure test. Additionally, an EIP with specified configurations is automatically assigned to each node. If the number of EIPs is less than the number of nodes, the EIPs are randomly bound to the nodes.
   - In the **Advanced Settings (Optional)** area, set **Kubernetes Node Name** to **Node private IP**. If you select **Cloud server name**, the node cannot be managed.
   - To enhance the security of a CCE node, see **Configuration Suggestions on CCE Node Security**.
6. Click **Next: Confirm**. Review the node configurations, select the disclaimer, and click **Submit**. After the node is created, return to the CodeArts PerfTest console.

Step 7 Create a resource group.

Choose **Resource Groups** in the navigation pane, and click **Create Private Resource Group**.

Configure the parameters by referring to **Table 5-3**.

Table 5-3 Creating a private resource group

Parameter	Description
Resource Group Name	Name of the private resource group to be created.
Debugging Cluster	Select a CCE cluster from the drop-down list.
Debug Node	Node used for debugging. It cannot be changed after the resource group is created.

Parameter	Description
Execution Node	Target machine that a pressure test will be performed on and can provide performance data during testing.

Click **Create**.

----End

5.3 Managing Private Resource Groups

After a private resource group is created, you can perform the following operations on it.

Viewing Information About a Private Resource Group

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **Resource Groups**.

Step 3 In a resource group card, you can view the information listed in the following table.

Table 5-4 Private resource group information

Name	Description
Private Resource Group Name	Name entered when you were creating a private resource group.
Private Resource Group Status	Running Abnormal Upgradable Deploying
Actuators	In-use/Total Actuators A single actuator supports up to 5,000 concurrent users for HTTP/HTTPS/WebSocket/MQTT or 1,000 for JMeter/HLS/RTMP/HTTP-FLV.
Current Concurrency	Sum of concurrent users of all node clusters in the current private resource group.
Debug Node	Debugging nodes selected when you were creating a private resource group.
Execution Node	Execution nodes of all node clusters in a resource group. Click **More** to view all execution nodes of the resource group.
Created At	Time when the private resource group was created.

Name	Description
Modified At	Last time when the private resource group was modified.

----End

Custom JMeter Installation Package

To switch the JMeter version, customize the JMeter installation package to meet service requirements.

Step 1 Download **JMeter installation package** of the required version.

Step 2 In the resource group list, click **Customization JMeter Installation package**, import the installation package, and wait until a message is displayed indicating that the import is successful. The JMeter installation package must be in .tgz, .tar.gz, or .zip format and smaller than 100 MB. The version must be 5.2.1 or later. Ensure that the installation package can be executable locally.

When uploading the custom JMeter installation package, pay attention to the following:

- Use a package from the Apache official website or a package with the same directory structure as the Apache package.
- Third-party plug-ins contained in the package will affect the JMeter engine of CodeArts PerfTest after being uploaded.

Step 3 (Optional) Delete the JMeter installation package.

If a message is displayed indicating that the upload fails, or you need to upload the JMeter installation package again, click to delete the current package.

----End

Adding a Cluster

When you create a private resource group, a node cluster is selected. If many clusters or clusters in different regions are required to run a pressure test task, add clusters and configure the pressure percent to allocate the task to different clusters.

Step 1 In the resource group card, click **Resource Adjust**.

Step 2 Add a cluster.

Click **Add Cluster**, set the basic information by referring to **Table 5-5**, and click **OK**.

Table 5-5 Adding a cluster

Parameter	Description
Region	Select the region of the created CCE node cluster from the drop-down list. The region can be different from the current CodeArts PerfTest region.
Node Cluster	Select a CCE cluster from the drop-down list.
Execution Node	Select a pressure target machine that provides performance data during a pressure test. If the region of the new cluster is different from the current CodeArts PerfTest region, use a cross-region node. Bind an EIP to the node or use NAT to allow the node to access the public network.

You can repeat this step to add more clusters.

Step 3 Configure the pressure percent.

After clusters are added, you can view all node clusters of the current resource group in the **Resource Adjust** dialog box, and adjust the **Pressure Percent** of different clusters.

Step 4 (Optional) Delete a node cluster.

In the **Resource Adjust** dialog box, click **Delete**.

----End

Scaling a Node Cluster

Step 1 In the resource group card, click **Resource Adjust**.

Step 2 Scale a node cluster.

- Scale out the resource group: Click **Scale**, select existing nodes and new nodes in the **Select Execution Node** dialog box, and click **OK**.
- Scale in the resource group: Click **Scale** under the resource group, select the execution nodes you want to keep, and click **OK**.

----End

Upgrading a Private Resource Group

Step 1 In the resource group list, click **Upgrade** under the desired resource group.

Step 2 Upgrade the resource group as prompted.

----End

Deleting a Private Resource Group

Step 1 In the resource group list, click **Delete** under the desired resource group.

Step 2 Click **OK**.

NOTE

When you delete a resource group, nodes will not be used in pressure tests again instead of being deleted. To delete a node permanently, go to the CCE console. For details, see **Deleting a Cluster**.

----End

6 PerfTest Project Management

6.1 Creating a PerfTest Project

CodeArts PerfTest helps you manage test projects. Transaction modes, test cases, test tasks, global variables, real-time reports, intelligent analysis results, and offline reports are shared in a test project. You can create different test projects for different test programs.

You can customize or use a template to create a test project. CodeArts PerfTest defines templates for the following scenarios:

- **All-in-one network office system scene**: You can quickly construct a pressure model by simulating system access requests to detect service performance bottlenecks under different pressure models and prevent service breakdown.
- **Stream scene**: Supports streaming push and pull with common streaming protocols. Construct corresponding pressure test scenarios and simulate typical audio and video scenarios to perform performance pressure tests to identify risks in advance.
- **Seckill scene**: Simulate a large number of users performing operations on offerings at a specified time point to check the reliability of e-mall services when the access pressure increases exponentially.
- **E-commerce scene**: Simulate a large number of merchants logging in to the e-commerce system and perform operations such as viewing and maintaining offering information. Verify that related operations are not interrupted and services can run properly.
- **Quick Create**: Quickly create customized pressure test scenarios and orchestrate test cases on the test case page, greatly improving test case design efficiency.

Creating a Test Project Using a Template

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **Dashboard**.

Step 3 Select a template to automatically create test tasks. CodeArts PerfTest provides the following templates:

- **All-in-one Scene**
- **Stream Scene**
- **Seckill Scene**
- **E-commerce Scene**
- **Quick Create**

You can modify a test case as required. For details, see **PerfTest Case Management**.

----End

Manually Creating a Test Project

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click **Create Test Project**.

Step 4 Configure the parameters by referring to **Table 6-1**.

Table 6-1 Creating a test project

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Project Name</td><td>Name of a new test project.
The project name can contain a maximum of 128 bytes, including letters, digits, and the following special characters: _/.,|`'~;:-+=×...!@#$%^&*?()<>{}[]¥"</td></tr>
<tr><td>Description</td><td>Description of the new test project. Enter up to 50 characters.</td></tr>
</table>

Step 5 When the configuration is complete, click **OK**.

After creating a test project, you can add a test case for the test project. For details, see **PerfTest Case Management**.

----End

6.2 Managing PerfTest Projects

After a test project is created, you can modify, delete, import or export it.

Modifying a Test Project

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

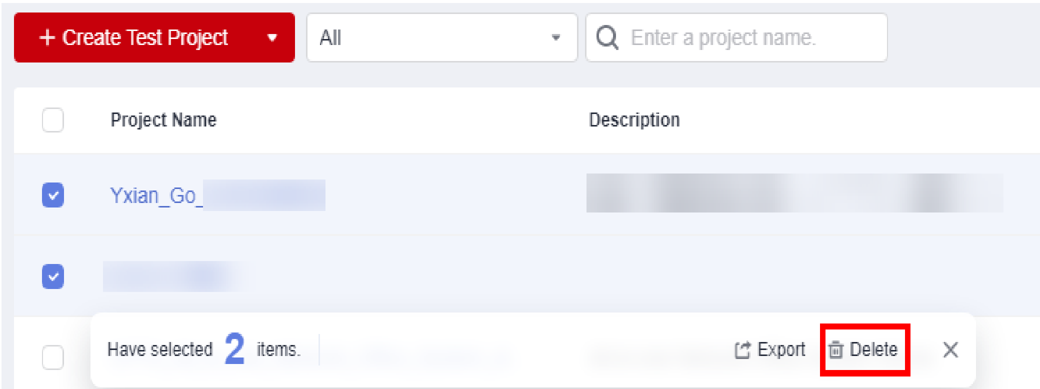
Step 3 In the PerfTest project list, locate the target project, click in the **Operation** column, enter a new project name or description, and click **OK**.

----End

Deleting Test Projects

Step 1 In the PerfTest project list, locate the target project, choose > **Delete Project** in the **Operation** column, and delete the project as prompted.

Step 2 Alternatively, select one or more test projects, click **Delete**, and delete them as prompted.



----End

Modifying Tags

Step 1 In the PerfTest project list, locate the target project, choose > **Modify Tag** in the **Operation** column. The **Resource Tags** tab page is displayed.

Step 2 Modify tags by referring to **Configuring Resource Tags for a PerfTest Project**.

----End

Exporting Test Projects

To streamline operations, you do not need to write a test project from scratch. You can find an existing project similar to your service model, export it, make the necessary modifications, and then import the updated project.

Step 1 Select one or more projects to be exported from the PerfTest project list, and click **Export**. Then save the exported test project file **ProjectList(*Serial number*).json** to the local host.

- If SLA rules have been configured for a project, they will not be included in the export.
- If a file with global variables has been uploaded to a project, this file will not be exported with the project.
- If a test project contains encrypted global variables, the project cannot be imported after being exported.

- Projects that use the MQTT protocol cannot be exported.

Step 2 Modify parameters and the project name (**the name must be unique**) in the file as required, and save the file. Do not change the format of the **ProjectList(***Serial number***).json** file. Altering the format may cause the subsequent import to fail.

----End

Importing a Test Project

Test projects of PerfTest and JMeter can be imported to CodeArts PerfTest.

Step 1 On the **PerfTest Projects** page, click **Import**.

Step 2 Import a PerfTest project. Currently, projects that use the MQTT protocol cannot be imported.

1. Set **Project Type** to **PerfTest project**.
2. Click **Select File**, select a project file in JSON format, and click **Import**.

   The name of the imported project must be unique in CodeArts PerfTest. Otherwise, the project cannot be imported.

Step 3 Import a JMeter project.

1. Set **Project Type** to **JMeter project**.
2. Click **Select File**, select a project file in **.jmx** format, and click **Import**.

Pay attention to the following points when importing a JMeter project:

- Ensure that the fields in the JMeter script are supported in CodeArts PerfTest. For details, see **Mapping Between JMeter and PerfTest Fields**.
- JMeter parses the following contents: thread group, HTTP request, HTTP header manager, regular expression extractor, user parameter, user-defined variable, default HTTP request value, and fixed timer.
- Random variables are not supported.
- To ensure the accuracy of data import, do not use variables with the same name.
- A JMeter script cannot exceed 5 MB.

Step 4 After the project is imported, click **Close**. You can modify the test project by referring to **Modifying a Test Project** and **Managing Transaction Requests**.

----End

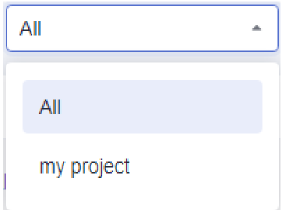
Viewing a Test Project

You can view the existing projects.

Step 1 On the **PerfTest Projects** page, enter a test project name in the search box to search for the target test project.

Step 2 You can also filter test projects by project scope. Click **Figure 6-1** and select the target test project.

Figure 6-1 Scope drop-down list



- **All**: displays all projects within the current user's permission scope, including projects created by subaccounts.
- **my project**: displays only projects created by the current account.

----End

6.3 Configuring Intelligent Analysis for PerfTest Projects

Intelligent analysis allows you to add analysis objects to your created test projects. Intelligent analysis is an enhanced capability. You can choose whether or not to configure it based on your requirements.

Currently, CodeArts PerfTest in the AP-Singapore region has been interconnected with APM 2.0, and the APM 1.0 configuration in CodeArts PerfTest is invalid. Use APM 2.0 when setting intelligent analysis.

Prerequisites

- APM has been enabled, a probe has been deployed, and an application has been generated. For details about how to deploy an APM probe, see **Monitoring Java Applications**.
- Application Operations Management (AOM) has been enabled, and a probe has been deployed. For details about AOM Agent deployment, see **Host Monitoring**.

Creating Analysis Objects

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the PerfTest project to which analysis objects are to be added to go to the details page.

Step 4 On the **Intelligent Analysis** tab page, click **Add Analysis Object**.

Step 5 In the displayed dialog box, the system automatically generates the analysis object name, but you can change it. Select the analysis object to be added. The analysis object can be either an application group, that is **Application Monitor (APM)**, or a node, that is **Node Monitor (AOM)**. The application group is the monitoring group that is set when APM probes are installed on the application server. The node is the VM where AOM probes are installed.

Step 6 Click **OK** after the configuration.

Step 7 Edit or delete analysis objects.

- Edit an analysis object: On the **Intelligent Analysis** tab page, click next to the analysis object to modify and edit it as prompted.

  If an analysis object is associated with a test task, once its parameters are edited, parameters of its associated object in the test task are also changed.

- Delete an analysis object: On the **Intelligent Analysis** tab page, click next to the analysis object to delete and delete it as prompted.

  Analysis objects cannot be deleted if they are being used in a test task.

----End

6.4 Configuring Resource Tags for a PerfTest Project

You can add resource tags to created PerfTest projects. Each PerfTest project can be regarded as a resource. You can tag resources under your account for classification. In scenarios such as cost management, you can quickly view details about a type of resources by tag. Resource tags are an enhanced capability. You can choose whether or not to configure them based on your requirements.

Configuring Resource Tags

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 On the **Resource Tags** tab, click **Add Tag**. Select a tag key, and a new key-value pair text box is displayed. You can add multiple tags at a time.

- A test project can have a maximum of 20 tags. A tag key must be unique and can have only one value.
- Tag keys of different test projects can be invoked by each other. For example, add tag keys **a** and **b** to Test Project 1. When you add tags to Test Project 2, **a** and **b** will be automatically displayed in the tag key drop-down list.

Step 5 Click **OK** after the configuration.

Step 6 You can edit and delete tags.

- Editing a tag: On the **Resource Tags** tab, click next to the desired tag, enter another value, and click **OK**.
- Deleting a tag: On the **Resource Tags** tab, click next to the desired tag, and click **OK**.

----End

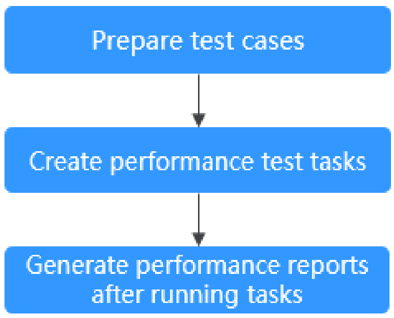
7 PerfTest Case Management

7.1 Overview

A PerfTest project consists of test cases, test tasks, and performance reports. Transactions and intelligent analysis are enhanced capabilities.

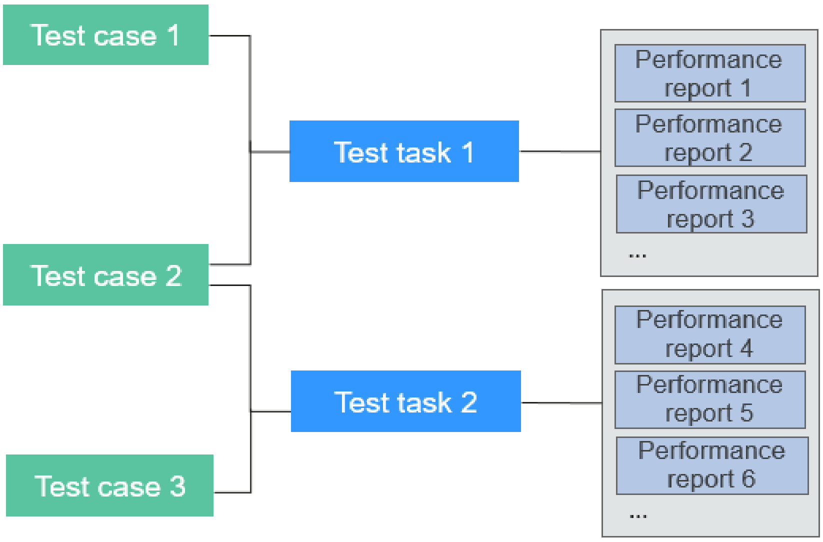
Test cases, test tasks, and performance reports are the three main steps of performance testing.

Figure 7-1 Performance testing process



A test case is a test model established based on a performance pressure test scenario. A test task is a performance test based on defined test models. A performance report is the execution result of a test task and displays the performance indicators of the tested system in high-concurrency scenarios.

Figure 7-2 Relationships between test cases, test tasks, and performance reports



7.2 Managing PerfTest Case Directories

A test case directory is a display structure for organizing and managing test cases in a test project.

Prerequisites

A **PerfTest project** has been created.

Creating a Test Case Directory

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

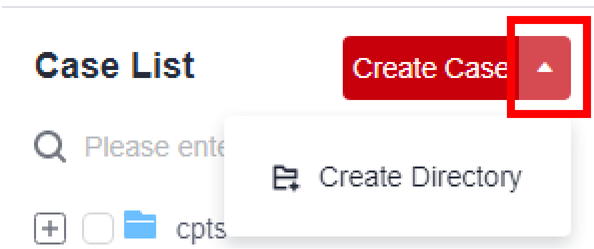
Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 On the **Cases** tab, create a directory.

- Create a case directory in the root directory.

  Click the arrow on the right of **Create Case**. Click **Create Directory** from the drop-down list and enter the directory name.

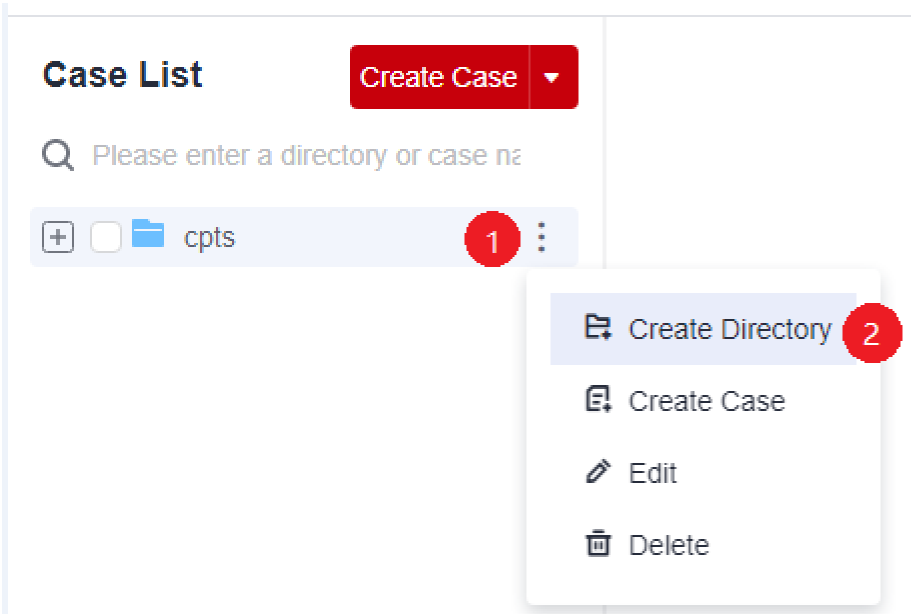
  Figure 7-3 Creating a case directory in the root directory

- Create a subdirectory in an existing directory.

  Click next to an existing directory, choose **Create Directory** from the drop-down list, and enter the directory name.

  Figure 7-4 Creating a subdirectory in an existing directory



----End

Modifying a Test Case Directory

Step 1 On the **Cases** tab, move the cursor to the directory to be modified.

Step 2 Click and choose **Edit**.

Step 3 Enter the new directory name in the text box and click another position on the page. The directory name is automatically saved.

----End

Deleting a Test Case Directory

Step 1 On the **Cases** tab, move the cursor to the test case directory to be deleted.

Step 2 Click and choose **Delete**.

Step 3 Click **OK**.

----End

7.3 Creating a PerfTest Case

A test case is a test model established based on a performance pressure test scenario.

Prerequisites

- A **PerfTest project** has been created.
- A **test case directory** has been created.

Creating a PerfTest Case

Step 1 **Log in to the CodeArts PerfTest console**.

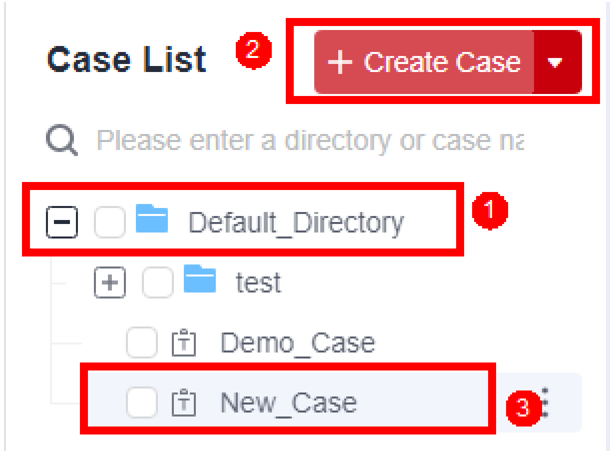
Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 On the left of the **Cases** tab, view the structure of the created case directories. You can create a case in either of the following ways:

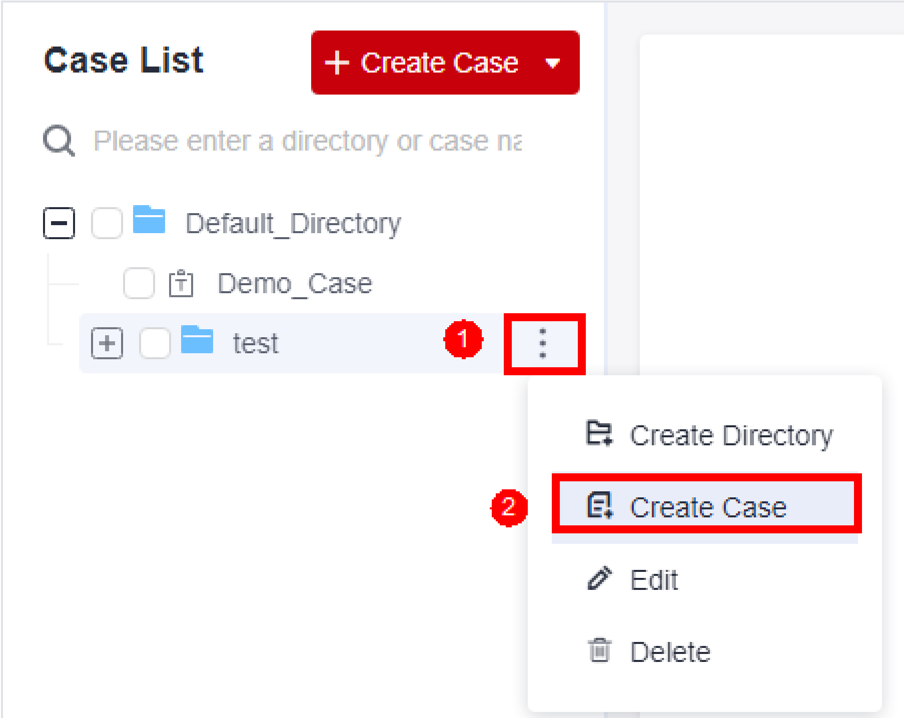
- To create a case in the current directory, click **Create Case**.

Figure 7-5 Creating a case in the current directory



- To create a case in a specified directory, move the cursor to the directory, click , and choose **Create Case**.

Figure 7-6 Creating a case in the specified directory



Step 5 Click **Save**.

----End

7.4 Configuring a PerfTest Case

7.4.1 Filling in PerfTest Case Details

Case details describe the background, preconditions and test steps of a test case to record its basic information for reuse. Case details are optional.

Filling in PerfTest Case Details

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, click the test case you want to configure.

Step 5 Click **Case Detail** on the right of the page. On the **Case Detail** tab, enter description, precondition, and procedure for the case, and set basic information by referring to **Table 7-1**.

Table 7-1 Basic case information

Parameter	Description
Status	Status of a test case, such as **Creating**, **Designing**, **Testing**, and **Finished**.
Case ID	ID of the test case.
Case Level	● **L0**: verification of underlying functions. Each module should have 10 to 20 test cases. L0 test cases account for 5% of all test cases. ● **L1**: basic function verification for inherited features or before iteration acceptance. L1 test cases account for 20% of all cases. ● **L2**: important feature verification for manual tests in non-regression versions. L2 test cases account for 60% of all cases. ● **L3**: 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 cases. ● **L4**: verification of special input, scenarios, and threshold conditions. L4 test cases account for less than 5% of all cases.
Tags	Custom case tags, which are used to identify the test case. ● Enter a tag: Click the text box, enter a tag, and press the **Space** or **Enter** key to generate the tag. Each case can have a maximum of 10 tags. ● Delete a tag: Click the text box and click next to a generated tag to delete it.
Create Time	Time when a test case was created. The value is automatically generated after the test case is saved.
Modify Time	Time when a test case was last modified. The value is automatically generated after the test case is saved.

Step 6 After configuring the case details, click **Save**.

----End

7.4.2 Configuring a PerfTest Case Script

After a test case is created, you can modify it as required.

Creating a Case Script

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, click the test case you want to configure.

Step 5 On the right of the page, click **Case Script** and choose **Pre-Steps** or **Steps**.

Pre-Steps are executed only once in each execution thread. This configuration is optional.

Steps are executor of a test case, which needs to be configured.

Step 6 Add concurrency weight controllers, throughput controllers, requests, transactions, data sets, cycle controllers, condition judgments, and rendezvous points as required.

You can add both requests and transactions, or add either of them. In practice, data sets, cycle controllers, condition judgments, and rendezvous points are used together with requests.

- For details about how to add requests, see **Adding PerfTest Request Information (Packet)**, **Adding PerfTest Request Information (Think Time)**, **Adding PerfTest Request Information (Response Extraction)**, and **Adding PerfTest Request Information (Checkpoint)**.
- Add a transaction. A transaction mode already exists on the **Transactions** tab page. Click **Add Transaction**, select a transaction name, and click **OK**. For details about how to create a transaction, see **Creating a PerfTest Transaction and Adding a Transaction Request**.
- For details about how to add data sets, cycle controllers, condition judgments, or rendezvous points, see **Adding a PerfTest Data Instruction, Cycle Controller, Condition Judgment, or Rendezvous Point**.
- For details about how to add concurrency weight controllers or throughput controllers, see **Adding PerfTest Concurrency Weight and Throughput Controllers**.

Step 7 Click **Save**.

----End

Skipping a Request Upon Failure

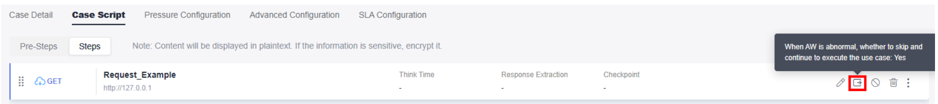
When this configuration is enabled, if a request fails during test case execution, the execution will not be terminated, and subsequent requests will proceed. This setting is necessary when you want the failure of a specific request not to impact the execution of critical subsequent steps.

Step 1 In the case list on the left of the **Cases** tab, click the test case you want to configure.

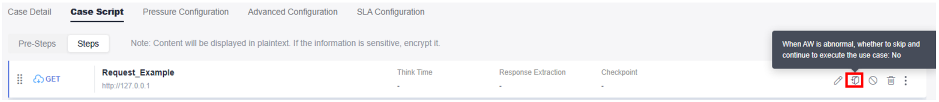
Step 2 On the **Case Script** tab, click the desired request.

Step 3 Click the second icon ( ) on the right of the request to configure the skip setting. The default value is **No**.

- **Yes**: Skip the failed request. Test case execution will continue without interruption.



- **No** (default): The failed request is not skipped. Request failure will terminate the overall case execution.

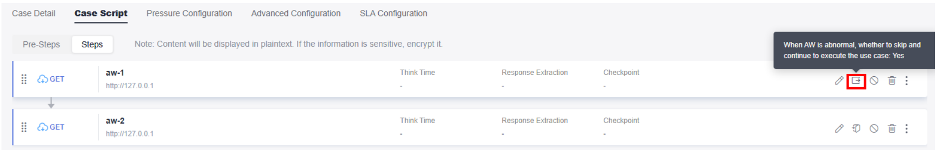


----End

Example

If request **aw-1** is configured to be skipped upon failure, it will be skipped when it fails, and the subsequent request **aw-2** will still be executed.

Figure 7-7 Skipping a request upon failure



Retrying a Request Upon Failure

This feature allows the system to automatically retry a failed request during test case execution based on configured parameters, including the retry times and retry interval. Configuring retries provides fault tolerance for requests that may fail due to transient issues like network instability or performance fluctuations.

Step 1 In the case list on the left of the **Cases** tab, click the test case you want to configure.

Step 2 On the **Case Script** tab, click the desired request.

Step 3 Click the **Checkpoint** tab and toggle on **Enable Result Check** to access the retry configuration settings.

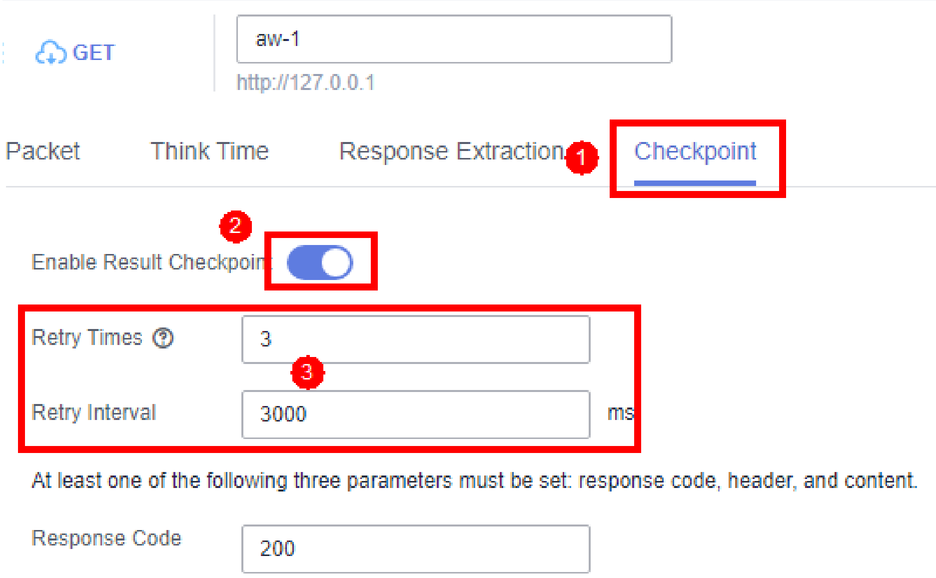
----End

Example

Request **aw-1** is configured to be retried upon failure, but not to be skipped upon failure.

- If **aw-1** executes successfully on the first attempt, request **aw-2** will be executed immediately without retrying **aw-1**.
- If **aw-1** fails, it will be retried three times at an interval of 3,000 ms (based on the checkpoint settings).
  - If **aw-1** is successful during any of the three retries, request **aw-2** will be executed.
  - If all three retries fail, request **aw-1** is marked as failed, and the aw-2 request execution will be interrupted.

Figure 7-8 Checkpoint configured for the aw-1 request



Disabling/Enabling a Case Request

If you want to retain a request of a case but do not want to use it, you can disable it. If you want to resume the use of the request, you can enable it again.

Step 1 In the case list on the left of the **Cases** tab, click the test case you want to configure.

Step 2 Disable a case request.

On the **Case Script** tab, click next to the desired request to disable it. Other requests are not affected.

Step 3 Enable a case request.

On the **Case Script** tab, click next to the desired request to enable it. Other requests are not affected.

----End

Deleting a Case Request

Step 1 In the case list on the left of the **Cases** tab, click the test case you want to configure.

Step 2 On the **Case Script** tab, locate the request to be deleted and click .

Step 3 Confirm the deletion as prompted.

----End

Copying/Pasting a Case Request

Step 1 In the case list on the left of the **Cases** tab, click the test case you want to configure.

Step 2 On the **Case Script** tab, locate the request to be copied and click ⋮ > **Copy**. "The copy is successful!" is displayed in the upper right corner of the page.

Step 3 Select a request in a test case of any PerfTest project, click ⋮ next to the request, and select **paste to front** or **paste to behind** to paste the copies request to the specified position.

----End

7.4.3 Adding PerfTest Request Information (Packet)

Packets are data blocks transmitted between HTTP/HTTPS/TCP/UDP/HLS/RTMP/WebSocket/HTTP-FLV/MQTT-based applications. Data blocks start with text metadata that describes the packet content and meaning. The text metadata is followed by optional data. Data blocks are transmitted among clients, servers, and agents.

Filling in a Packet

Packets refer to all click operations performed by users on the website interface. The process of sending a packet: A click operation is edited to a code stream complying with protocol specifications and carrying a user's request before the code stream is sent to a third party, leading to a correct or failed response. A correct response indicates that the operation is successful, and a failed response provides tips for rectifying the problem.

CodeArts PerfTest supports HTTP/HTTPS packets of the following request types: GET, POST, PATCH, PUT, and DELETE.

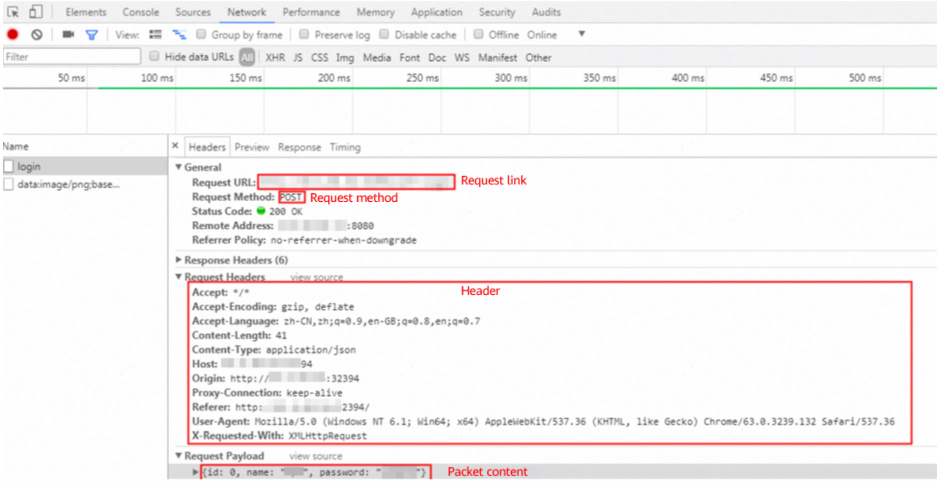
Before filling in a packet, you need to confirm the action of the request API. Taking queries as an example, query messages are GET requests. You can set the request mode to GET during configuration.

If a request involves various information input, you can press **F12** or use a packet capture tool (such as Wireshark) to check how the packet is requested, what the packet body format is, and how the packet is transmitted to third-party APIs. Then, fill in the packet in the pressure test based on your service requirements.

For standard HTTP/HTTPS packets, fill in the packet header based on the captured content. The packet body specifies the request content, which depends on the service to be tested. The body can be a game login request or a registration request. All packets can be edited for pressure tests as long as the request complies with the HTTP or HTTPS protocol. Generally, such a request uses the POST method. After this method is selected, the following information is displayed.

For example, to simulate the POST method of a login request, the corresponding request URL, request mode, header field, and packet content are shown in **Figure 7-9**.

Figure 7-9 Simulating a login request using the POST method



The preceding example is also applicable to PATCH, PUT, and DELETE methods. First, confirm the protocol type, request method, and request link of the application to be tested, and then confirm the content in a request.

Adding HTTP and HTTPS Packets

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, select the test case for which you want to add request information.

Step 5 On the **Case Script** tab, click **Add Request**.

Step 6 On the **Packet** tab, select **HTTP** or **HTTPS** for **Request Information**.

Step 7 Configure the parameters by referring to **Table 7-2**.

Table 7-2 HTTP and HTTPS packet parameters

Parameter	Description
Request mode	**GET**, **POST**, **PATCH**, **PUT**, or **DELETE**.
Request URL	URL for sending a request, for example, **http://***domain name***/***path*, or **http://***domain name***/path?key1=value1&key2=value2**. Variables can be inserted. For details, see **Inserting Variables in a PerfTest Request**.
Response Timeout (ms)	Timeout period for waiting for a response from the server when a request is sent. The response timeout ranges from 20 ms to 30 min. If this parameter is not set, the default response timeout interval (5,000 ms) is used.

Parameter	Description
Carried cookie	At least one request exists in the case. Set this parameter when adding new requests. ● **Obtain automatically**: Use the cookie set in the response. ● **Obtain manually**: Use the cookie in the header of the current request.
Automatic Redirection	Whether to redirect to the redirection address. The function is enabled by default, that is, the redirection address is automatically redirected to.
Request Parameter	Set the parameters in the request URL. Click **Add Request Parameter** and set **Name** and **Value**. Select **Url Encode** if necessary. By default, it is not selected. If it is selected, URL encoding is performed on the values of request parameters. If you do not need to add a request parameter, click **Delete** to delete it.
Request Header	Add headers and contents based on the headers to be verified or used by a pressure test server. Headers are not mandatory. Only user-defined headers are transparently transmitted to the pressure test server. For details about **Header**, see **Header Description**. ● Click **Add Header** and set **Header** and **Value**. If you do not need to add header information, click **Delete** to delete the packet header. ● You can click **Bulk Edit** to edit multiple headers at a time in the text box. You can click **Key-Value Edit** to return to the key-value pair editing mode. If the request mode is **POST** or **PUT** and **Header** is **Content-Type**, variables can be inserted, and **Value** can be one of the three following types: ● **Customize**: Enter a value in the text box. ● **application/x-www-form-urlencoded**: The request body is a key value pair that can be added. The value is text. ● **multipart/form-data**: The request body is a key-value pair that can be added. The value can be text or a file. If the value is a file, the body can be imported.

Parameter	Description
Body	The body of an entity contains a data block consisting of random data. Not all packets contain the body of an entity. If global variables have been set or local variables have been set for response extraction, variables can be used in the packet content. During a pressure test, variables in the packet content will be replaced with specified values. 1. Enter **$** in the request content input box. 2. In the **Insert Variable** dialog box, set parameters. For details, see **Inserting Variables in a PerfTest Request**. 3. Click **Select**.

Step 8 Click **Save**.

----End

Adding a TCP Packet

Step 1 On the **Packet** tab, select **TCP** for **Request Information**.

Step 2 Configure the parameters by referring to **Table 7-3**.

Table 7-3 TCP packet parameters

Parameter	Description
IP	IP address of the tested server to which requests are sent.
Port Number	Port number of the tested server to which requests are sent.
Connection Timeout	Timeout duration for the server's response after a connection is initiated. The connection timeout value ranges from 20 ms to 60s.
Response Timeout	Timeout duration for waiting for the server's response after a connection is established. The response timeout ranges from 20 ms to 30 min.
Connection Settings	- **Repeated use**: When a request response is complete, the connection remains and is used to send and receive the next request response. - **Close**: When a request response is complete, the connection is closed and re-established next time.

Parameter	Description
Return Settings	Used to judge whether a request response has been received. It is recommended that a unique end-of-text character be set. If multiple such characters are present in a response, the response is considered complete when the first character is received. As a result, the received response data may be incomplete. ● **Length of returned data**: length of the returned data, in bytes. When a response of this length is received, data receiving is complete. ● **End-of-text character**: ending mark of the returned data. When an end-of-text character is received, data receiving is complete.
Content Format	Select **String** or **Hexadecimal code stream data** based on the service request content of the tested server. A hexadecimal code stream can contain only numbers 0–9 and letters a–f. The total number of characters must be an even number. If global variables have been set or local variables have been set for response extraction, variables can be used in the packet content. During a pressure test, variables in the packet content will be replaced with specified values. 1. Enter **$** in the request content input box. 2. In the **Insert Variable** dialog box, set parameters. For details, see **Inserting Variables in a PerfTest Request**. 3. Click **Select**.

Step 3 Click **Save**.

----End

Adding a UDP Packet

Step 1 On the **Packet** tab, select **UDP** for **Request Information**.

Step 2 Configure the parameters by referring to **Table 7-4**.

Table 7-4 UDP packet parameters

Parameter	Description
IP	IP address of the tested server to which requests are sent.
Port Number	Port number of the tested server to which requests are sent.

Parameter	Description
Return Settings	Used to judge whether a request response has been received. It is recommended that a unique end-of-text character be set. If multiple such characters are present in a response, the response is considered complete when the first character is received. As a result, the received response data may be incomplete. ● **Length of returned data**: length of the returned data, in bytes. When a response of this length is received, data receiving is complete. ● **End-of-text character**: ending mark of the returned data. When an end-of-text character is received, data receiving is complete.
Content Format	Select **String** or **Hexadecimal code stream data** based on the service request content of the tested server. A hexadecimal code stream can contain only numbers 0–9 and letters a–f. The total number of characters must be an even number. If global variables have been set or local variables have been set for response extraction, variables can be used in the packet content. During a pressure test, variables in the packet content will be replaced with specified values. 1. Enter **$** in the request content input box. 2. In the **Insert Variable** dialog box, set parameters. For details, see **Inserting Variables in a PerfTest Request**. 3. Click **Select**.

Step 3 Click **Save**.

----End

Adding a WebSocket Packet

Step 1 On the **Packet** tab, select **WebSocket** for **Request Information**.

Step 2 Configure the parameters by referring to **Table 7-5**.

Table 7-5 WebSocket packet parameters

Parameter	Description
Request mode	• **Connect**: Establish a WebSocket connection to the pressure test website. • **Disconnect**: Close the connection to the pressure test website. • **Pong**: Send **pong** to the pressure test website. • **Ping&Pong**: Send **pong** to the pressure test website and expect to return **ping**. • **Request**: Send a request to the pressure test website and receive a response. • **ReadOnly**: Only receive information sent by the pressure test website. • **WriteOnly**: Only send information to the pressure test website.
Request connection address	URL of the WebSocket connection, for example, **ws://***domain name*/*path*. Encrypted requests are supported, such as **wss://***domain name*/*path*.
Connection Timeout	Set this parameter when the request mode is **Connect**. Timeout duration for the server's response after a connection is initiated. The connection timeout value ranges from 20 ms to 60s.
Status Code	Set this parameter when the request mode is **Disconnect**. Status code for closing a connection. The default value is **1,000**. The value range is 0–4,999.
Request Type	Set this parameter when the request mode is **Request** or **WriteOnly**. Select **Text** or **Binary** based on the service request content of the tested server. **Binary** indicates hexadecimal.
Body	Set this parameter when the request mode is **Request** or **WriteOnly**. Enter the packet content of request information based on the specified request type. When the request content is text, you can enter variables. For details, see **Inserting Variables in a PerfTest Request**.

Parameter	Description
Response Type	Set this parameter when the request mode is **Request** or **ReadOnly**. Select **Text** or **Binary** based on the service response content of the tested server. **Binary** indicates hexadecimal.
Response Timeout	Timeout duration for waiting for the server's response after a connection is established. The response timeout ranges from 20 ms to 30 min.
Request Header	Add headers and contents based on the headers to be verified or used by a pressure test server. Headers are not mandatory. Only user-defined headers are transparently transmitted to the pressure test server. For details about **Header**, see **Header Description**. ● Click **Add Header** and set **Header** and **Value**. If you do not need to add header information, click **Delete** to delete the packet header. ● You can click **Bulk Edit** to edit multiple headers at a time in the text box. You can click **Key-Value Edit** to return to the key-value pair editing mode.

Step 3 Click **Save**.

----End

Adding an MQTT Packet

Step 1 On the **Packet** tab, select **MQTT** for **Request Information**.

Step 2 Configure the parameters by referring to **Table 7-6**.

Table 7-6 MQTT packet parameters

Parameter	Description
Request mode	● **Connect**: Establish an MQTT connection to the pressure test website. ● **Publish**: Publish a message. ● **Subscribe**: Subscribe to messages. ● **Disconnect**: Close the MQTT connection to the pressure test website.
When the request mode is **Connect**, set the following parameters.	
MQTT Version	MQTT version. Currently, 3.1 and 3.1.1 are supported.

Parameter	Description
Transport Address	MQTT server address, which supports the TCP, SSL, WebSocket (WS), and WebSocket Secure (WSS) protocols. Domain names cannot be bound to MQTT transmission addresses.
Timeout interval	Timeout interval for the client to perform operations such as establishing a connection and sending a message.
Client ID	MQTT client ID. You can set whether to add a random suffix to the client ID by enabling or disabling **Add a random suffix**.
Username	Username set for the MQTT client. This parameter is sensitive. Set it using global variables in sensitive parameter mode.
Password	Password set for the MQTT client. This parameter is sensitive. Set it using global variables in sensitive parameter mode.
Connection keepalive	Active heartbeat interval, in seconds.
Clear Sessions	● If this option is toggled on, the reserved session information of the last connection will be cleared when a connection is established between the client and the server. ● If this option is toggled off, the reserved session information of the last connection will not be cleared when a connection is established between the client and the server.

Parameter	Description
Add Will	● You can toggle on this option to add a will. In this way, when the client is disconnected abnormally, the MQTT agent automatically publishes the will message to other subscribers. Set the following parameters when adding a will: – **Theme**: topic of the will. – **Type**: **Text** or **Hex**. You can also set whether to add a timestamp. – **Content**: content of the will. – **QoS Level**: quality of service (QoS) level, which is used to describe the reliability of message transmission. **Send at most once**: applicable to real-time data transmission. **Send at least once**: applicable to applications that require reliability but allow a slight delay. **Only once**: applicable to applications that require high reliability and allow a long delay. – **Keep your will**: If this option is toggled on, the subscribing client receives the set will message immediately after subscription. If this option is toggled off, the subscribing client receives the will message only after the client that publishes the will message is disconnected unexpectedly. ● If you toggle off **Add Will**, no will message will be sent.
If the request mode is **Publish**, set the following parameters:	
Theme	Topic of the message to be published.
QoS Level	QoS level, which is used to describe the reliability of message transmission. **Send at most once**: applicable to real-time data transmission. **Send at least once**: applicable to applications that require reliability but allow a slight delay. **Only once**: applicable to applications that require high reliability and allow a long delay.
Request Type	**Text** or **Hex**. You can also set whether to add a timestamp.
Request Content	Detailed content of the request.
Timeout interval	Timeout interval for the client to send a message.
Reserved Message	● If this option is toggled on, messages published when the client is disconnected are stored on the server. ● If this option is toggled off, messages published when the client is disconnected are cleared.

Parameter	Description
When the request mode is **Subscribe**, set the following parameters.	
Subscribe to Topic	Topic of the subscribed message, which must match the topic of the published message.
QoS Level	QoS level, which is used to describe the reliability of message transmission. **Send at most once**: applicable to real-time data transmission. **Send at least once**: applicable to applications that require reliability but allow a slight delay. **Only once**: applicable to applications that require high reliability and allow a long delay.
Response Type	**Text** or **Hex**. You can also set whether to add a timestamp.
Timeout interval	Timeout interval of the subscription topic.
End Condition	When this condition is met, the subscription ends. ● **Maximum triggering time**: The subscription ends when the time reaches the upper limit. ● **Maximum Number of Received Messages Triggered**: The subscription ends when the number of received messages reaches the upper limit.
If the request mode is **Disconnect**, you do not need to set parameters.	

Step 3 Click **Save**.

----End

Adding an HLS Packet

Step 1 On the **Packet** tab, select **HLS** for **Request Information**.

Step 2 Configure the parameters by referring to **Table 7-7**.

Table 7-7 HLS packet parameters

Parameter	Description
Streaming media address	Address of the video source, for example, **http://***domain name*/*path*.
Play Duration (s)	Duration of simulated playback. During case debugging, the playback duration is set to 3 seconds.
Retry Delay (ms)	Interval for obtaining new data in live streaming when the video playback source does not have new data.

Parameter	Description
Retry Times	Maximum number of attempts to obtain new data in live streaming when the video playback source does not have new data. If the number of attempts exceeds the maximum, the system determines that the attempt fails.

Step 3 Click **Save**.

----End

Adding an RTMP Packet

Step 1 On the **Packet** tab, select **RTMP** for **Request Information**.

Step 2 Configure the parameters by referring to **Table 7-8**.

Table 7-8 RTMP packet parameters

Parameter	Description
Request Mode	• **Stream pushing**: a process of transmitting onsite video signals to the network. • **Stream pulling**: a process of pulling streaming media video files from a server using a specified address.
When **Request Mode** is **Stream pushing**, set the following parameters.	
Streaming server address	IP address of the tested streaming media server.
Port Number	Listening port number of the tested streaming media server. The default value is **1935** for RTMP.
App name	AppName in the ingest URL, that is, the path for storing live media files, for example, **live**.
Streaming name	StreamName in the ingest URL, which uniquely identifies a live stream, for example, **livestream**.
Push duration (s)	Duration for simulating stream pushing, in seconds.
Video source address	OBS address for storing video source files for simulating live stream pushing. Currently, only .flv files are supported. Non-standard H.265-encoded .flv files are not supported.
When **Request Mode** is **Stream pulling**, set the following parameters.	
Streaming server address	IP address of the tested streaming media server.

Parameter	Description
Port Number	Listening port number of the tested streaming media server. The default value is **1935** for RTMP.
App name	AppName in the streaming URL, that is, the path for storing live media files, for example, **live**.
Streaming name	StreamName in the streaming URL, which uniquely identifies a live stream, for example, **livestream**.
Play Duration (s)	Simulated duration of live stream watching, in seconds.

Step 3 Click **Save**.

----End

Adding an HTTP-FLV Packet

Step 1 On the **Packet** tab, select **HTTP-FLV** for **Request Information**.

Step 2 Configure the parameters by referring to **Table 7-9**.

Table 7-9 HTTP-FLV packet parameters

Parameter	Description
Streaming media address	Address of the video source, for example, **http://***{Video source IP address}***:8080/live/stream.flv**.
Play Duration (s)	Duration of simulated playback. During case debugging, the playback duration is set to 3 seconds.

Step 3 Click **Save**.

----End

7.4.4 Adding PerfTest Request Information (Think Time)

To better simulate user behavior, insert a waiting time between different operations. For example, when a user receives data from the server, the user may wait several seconds before viewing data and providing responses. This period of time is called think time.

Adding Think Time

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, select the test case for which you want to add request information.

Step 5 On the **Case Script** tab, click **Add Request**.

Step 6 On the **Think Time** tab page, set the basic information by referring to **Table 7-10**.

Table 7-10 Think time parameters

Parameter	Description
Enable Think Time	After think time is enabled, you may wait for several seconds before viewing the data received from the server, and then respond.
Duration (ms)	Think time refers to the waiting time between two consecutive operations of a user, such as the interval between login and search. For example, the response time for each case execution is 0.5s. ● To execute two case requests per second, do not add a case request whose type is **think time**. ● To execute only one case request per second, set **Duration** to 1s. If the think time is set to 1s and the response time is longer than 1s, the think time will not take effect, and pressure test requests are sent according to the response time.

Step 7 Click **Save**.

----End

7.4.5 Adding PerfTest Request Information (Response Extraction)

If multiple packets exist in the same case, use regular expressions or JSON to extract the output of the previous packet for the input of the next packet.

For example, in an e-commerce flash sale scenario, after you search for a product and purchase it, you can use the response extraction function to extract the product ID obtained from packet search and use the ID as the input parameter of the next purchase packet.

Notes and Constraints

- Cross-case extraction is not supported. That is, the output extracted from case A's packet cannot be used for the input of case B's packet.
- The variables from a response extraction are local variables.
- After an array is extracted from a response variable, the array can only be used as the input of the array variable in the next packet. Each field in the array cannot be used separately.

Adding Response Extraction

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, select the test case for which you want to add request information.

Step 5 On the **Case Script** tab, click **Add Request**.

Step 6 On the **Response Extraction** tab page, set the basic information by referring to **Table 7-11**.

Table 7-11 Response extraction parameters

Parameter	Description
Enable Response Extraction	After response extraction is enabled, if multiple packets exist in the same case, you can use regular expressions or JSON to extract the output of the previous packet for the input of the next packet.
Variable Name	Must be unique. The value of the response extraction is assigned to this variable.
Expected Value	Expected variable value of response extraction. If you enable this option, the extracted value is compared with the expected value. If they are different from each other, the URL verification will fail.
Extraction Range	Range of the content to be extracted. The packet content, header, and URL can be extracted using regular expressions.
Regular Expression	A regular expression specifies a type of logical expressions performed on strings. In a regular expression, you use certain predefined strings and a combination of these strings to form a rule string that is used to specify a specific filtering logic. Parentheses **()** indicate extraction and are used to enclose content to be extracted. The content in a pair of parentheses forms a sub-expression. A complete regular expression consists of two types of characters: special characters (also called meta characters) and literal or common text characters (such as letters, digits, and underscores). For details about meta characters, see **Regular Expression Metacharacters**.
Sequence Number of Matching Item	This parameter indicates the sequence number of the matched content extracted through a regular expression. This parameter cannot be set to **0**. Value range: positive integers

Parameter	Description
Expression Value	Value of the sub-expression that is parsed. The value is a natural number. ● Value **0** indicates that the entire regular expression is matched. ● Value **1** indicates that the first sub-expression of the regular expression is matched, that is, the content extracted by the first **()**. After extracting content through **Regular Expression** and **Sequence Number of Matching Item**, use **Expression Value** to obtain the final content.
Key Name	Enter the key name to be obtained. This parameter is valid only when **Extraction Range** is set to **Parameter values in .json files**. For example, to extract **v42** from {"key":{"key1":"v1","key2":{"key3":"v3"},"key4":[{"key41":"v41","key42":"v42"},{"key41":"v43","key42":"v44"}]}}, enter **key.key4[0].key42**.
Default Value	Value extracted when a regular expression match or JSON extraction fails.
Condition Expression	This parameter is used together with the **Key Name** to be obtained. For example, {"key":{"key1":"v1","key2":{"key3":"v3"},"key4":[{"key41":"v41","key42":"v42"},{"key41":"v43","key42":"v44"}]}}. If you want to describe when key42 is equal to v42, extract the target value **v41**, you can enter **key.key4[].key42=v42** for **Condition Expression**, and enter **key.key4[].key41** for **Key Name**.

Table 7-12 Common regular expressions

Regular Expression	Description	Example
(\d+)	Matches non-negative integers.	String: `bTivm2wu9jih1LBKR4osZGrjjl` Match results: 2 9 1 4
([A-Za-z]+)	Matches a string of letters.	String: `bTivm2wu9jih1LBKR4osZGrjjl`

Regular Expression	Description	Example
		Match results: bTivm wu jih LBKR osZGrjjl
([A-Za-z0-9]+)	Matches a string of digits and letters.	String: bTivm2wu9jih1LBKR4osZGrjjl:asdasd22
		Match results: bTivm2wu9jih1LBKR4osZGrjjl asdasd22
(\w+)	Matches a string of digits, letters, or underscores (_).	String: bTivm2wu9jih1LBKR4osZGrjjl:asdasd22
		Match results: bTivm2wu9jih1LBKR4osZGrjjl asdasd22
([\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+)	Matches email addresses.	String: bTivm2wu9jih1LBKR4osZGrjjl:abc@abc.com
		Match results: abc@abc.com

Step 7 (Optional) A request can contain one or more variables. If you need to add multiple variables for a request, click **Add Variable** and set related parameters.

Step 8 Click **Save**.

----End

Examples of Regular Expression Response Extraction

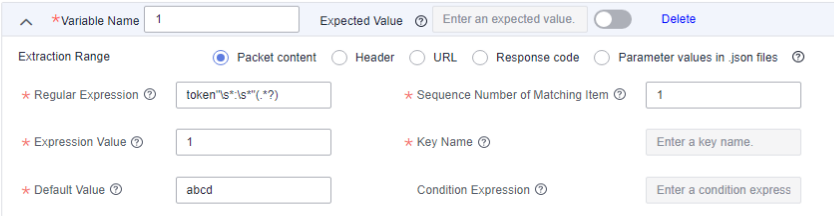
Example 1

The response content of the previous packet is as follows:


```
"baseInfo" : {
"mobilephone":"xxxxxxxxxxx",
"Telephone":"xxxxxxxxxxx",
"unitGuid":"xx",
"unitName":null,
"address":"xxx",
"gender" : 0,
"imageUrl" :null
},
"UserNotices":null
},
"msgId'" : "64xxxxxxxxxxxxxxxxxxxxxxxf5",
"isUsed" :"1"
"token":"eyxxxxxxxxxxxJ9.eyJzdW1101JYZMiLCJcUbdGUkIJezMT!!zz#z!
20TxxxxxxxxxxxxxxxxxS1611dYQISHTFKifVOsImd1aWQi0ilOODkiLCJleHAiOjEiMzIizNzY1MjZ9.myU5idiASM-1
1@EP7YQTfTsR_8zsq7?sbYJYoxfRsuf6OZhGL-
XWmjnvdaviGauhSdw16ImWOFEvbACSHMXGT1U0ijS5z6ezX@sZePruzFnvcIMgShF8xNPN6zVokQp-
uwbyS3W6NpZpDuwsjuiZ7DZTNpKoqCkGHwvPJrHBOrWFR_u6-FBbTiFiqdhQb95U-1gLiLvoZHY_rguzwyrZ-
leGRdCG_ZASreoWC-
uH)HnqltpgItrChWQToHQyxOABdMSbBSHhNctBBZHgQPMESqQQQTbBiPGvbQDprB7ZBFMUB_ShynS_evtyfE
```

```
ladGEddhOBn-fxxxxxxxxxxx"
},
```

To extract the token value, use the regular expression **"token"\s*:\s*"(.*?)"**. The configuration of response extraction is shown as follows:

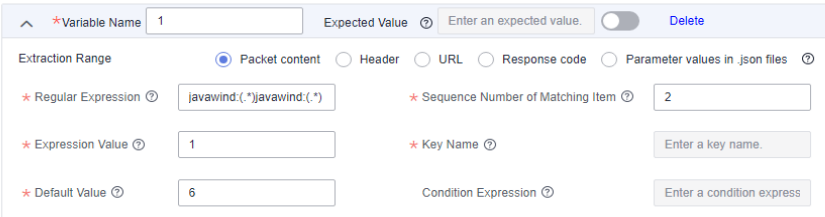


Example 2

The response content of the previous packet is as follows:

```
javawind:9javawind:12
javawind:16javawind:17
javawind:46javawind:22
```

To extract the value 16, see the following figure.

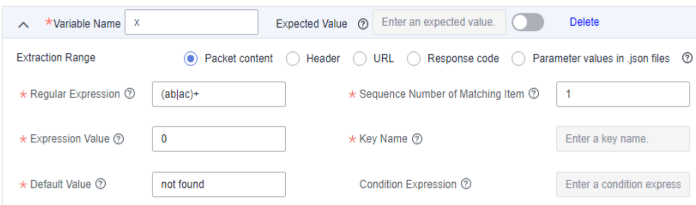


1. The following data is extracted using the regular expression **javawind:(.*)javawind:(.*)**.
   ```
   9 12
   16 17
   46 22
   ```
2. The following data is extracted using the second matching item.
   ```
   16 17
   ```
3. The following data is extracted using the first expression.
   ```
   16
   ```

Example 3

The packet content is **ababdacac**.

Response extraction settings:



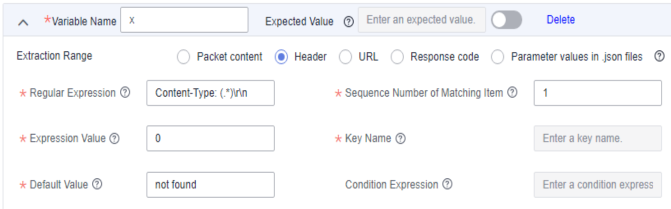
Regular Expression: **(ab|ac)+** indicates that multiple **ab** or **ac** values are matched.

Sequence Number of Matching Item: **1**, indicating that the matching item is matched by **ab**. **2**, indicating that the matching item is matched by **ac**. **1** may be used to obtain **abab** and a substring **ab**, and **2** may be used to obtain **acac** and a substring **ac**.

Expression Value: The value **0** indicates that the maximum matching string **abab** or **acac** is used. The value **1** indicates that the substring **ab** or **ac** is used.

Example 4

The extracted content is **Content-Type** in the HTTP response header.



Regular Expression: **Content-Type: (.*)\r\n**. Note that there is a space after the colon (:) and **\r\n** is at the end, which is generated based on HTTP specifications.

Sequence Number of Matching Item: **1**, because there is only one **(.*)**.

Expression Value: **1**, indicating that the expected value is obtained.

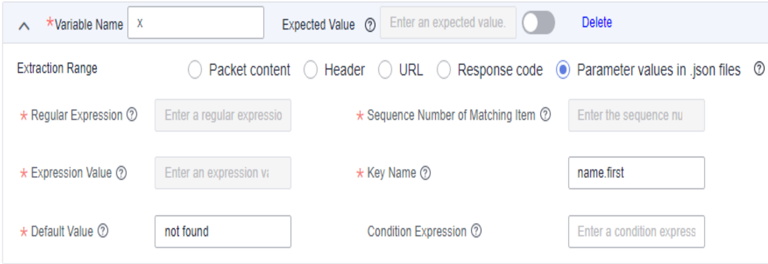
Examples of JSON Expression Response Extraction

The sample data is as follows:

```
{
   "name": {
          "first": "Tom",
          "last": "Anderson"
   },
   "age": 37,
   "children": ["Sara","Alex","Jack"],
   "fav.movie": "Deer Hunter",
   "friends": [{
          "first": "Dale",
          "last": "Murphy",
          "age": 44,
          "nets": ["ig","fb","tw"]
   },
   {
          "first": "Roger",
          "last": "Craig",
          "age": 68,
          "nets": ["fb","tw"]
   },
   {
          "first": "Jane",
          "last": "Murphy",
          "age": 47,
          "nets": ["ig","tw"]
   }]
}
```

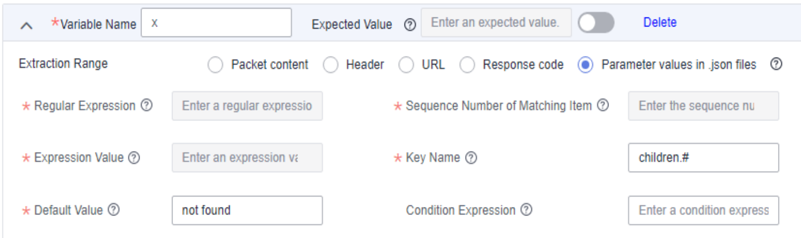
Example 1

To extract the **first name**, set **Key Name** to **name.first**.



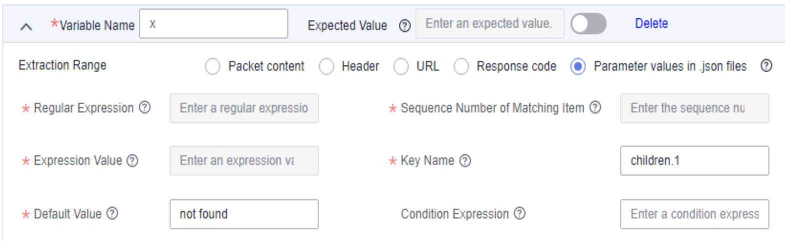
Example 2

To extract **the number of Tom's children**, set **Key Name** to **children.#**.



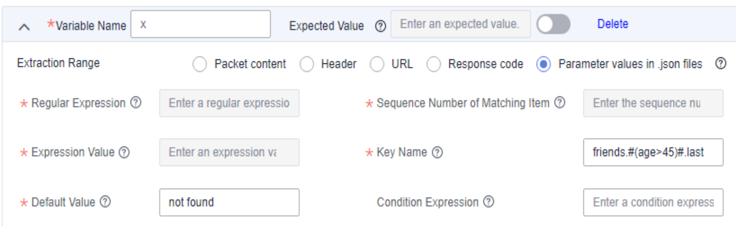
Example 3

To extract **the name of Tom's second child**, set **Key Name** to **children.1**. (Sequence number starts from 0.)



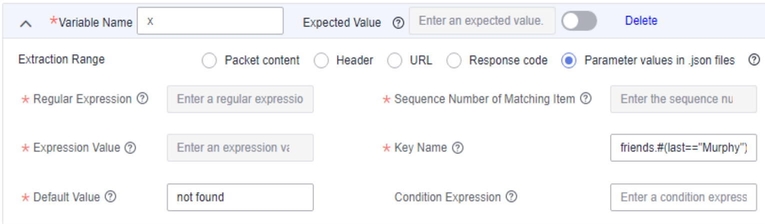
Example 4

To extract **the last names of Tom's friends older than 45**, set **Key Name** to **friends.#(age>45)#.last**.



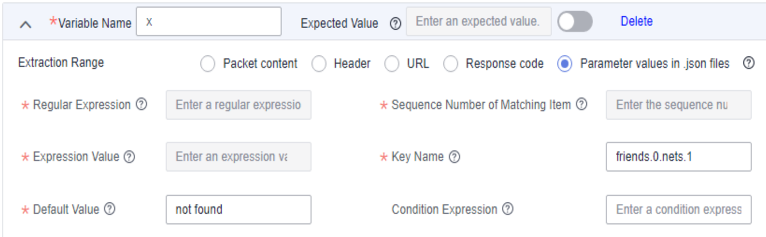
Example 5

To extract **the first name of Tom's friend whose last name is Murphy**, set **Key Name** to **friends.#(last=="Murphy")#.first**.



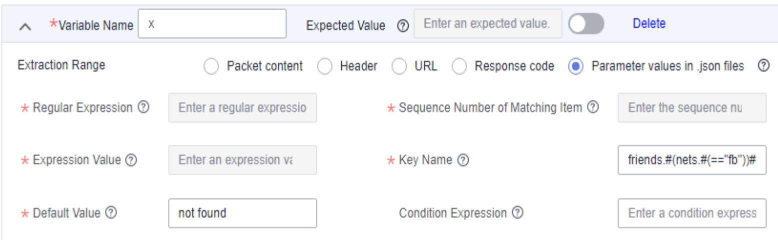
Example 6

To extract **the second nickname of Tom's first friend**, set **Key Name** to **friends.0.nets.1**. (Sequence number starts from 0.)



Example 7

To extract **the first name of Tom's friend whose nickname is fb**, set **Key Name** to **friends.#(nets.#(=="fb"))#.first**.



7.4.6 Adding PerfTest Request Information (Checkpoint)

Checkpoint uses customized verification information to check whether the content returned by a server is correct. For different protocol types, the checkpoint supports different contents. The HTTP and HTTPS protocols support the verification of response code, header, and content. The TCP, UDP, MQTT, and WebSocket protocols support only content verification.

Adding a Checkpoint

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, select the test case for which you want to add request information.

Step 5 On the **Case Script** tab, click **Add Request**.

Step 6 On the **Checkpoint** tab, set the parameters by referring to **Table 7-13**.

Table 7-13 Checkpoint parameters

Parameter	Description
Enable Result Checkpoint	After this option is enabled, you can check whether the content returned by a server is correct through customized verification information.
Retry Times	Number of times to retry a failed request until it succeeds. The value ranges from 0 to 100. Statistics such as response time, RPS, and success rate of retried requests are not included in the report. The report counts only the last data.
Retry Interval	Interval for retrying a failed request. The unit is ms. The value ranges from 0 to 3,600,000, that is, the maximum interval is 1 hour.
Response Code	Set this parameter only when the protocol type is set to **HTTP** or **HTTPS** on the **Packet** tab page. This parameter is the HTTP or HTTPS response status codes, including **1XX**, **2XX**, **3XX**, **4XX**, and **5XX**, carried in a packet.
Header	Set this parameter only when the protocol type is set to **HTTP** or **HTTPS** on the **Packet** tab page. Headers of HTTP and HTTPS. 1. Click **Add Header**. 2. Enter a **Header Name**. For details, see **Header Description**. 3. Set check conditions. 4. Set contents.
Response Content	Body parts of HTTP, HTTPS, TCP, UDP, MQTT, and WebSocket, namely the load parts of the requests and responses. 1. Click **Add Content**. 2. Set check conditions.
Condition Rule	• **AND**: The verification is successful only when all the check conditions are met. • **OR**: The verification is successful when any check condition is met.

Step 7 Click **Save**.

----End

7.4.7 Adding a PerfTest Data Instruction, Cycle Controller, Condition Judgment, or Rendezvous Point

Adding a Data Instruction

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, select the test case for which you want to add a data instruction.

Step 5 On the **Case Script** tab, choose **Add More** > **Data Set**.

Step 6 Click **Add Param**, and configure **Name**, **Logic**, and **Mode**.

- **Name**

  Name of the parameter to which the parameter logic result will be assigned as the parameter value.

- **Logic**

  a. To assign a constant value to a parameter, enter a character string, for example, **1.23** or **abc**.

  b. To assign a variable value to a parameter, enter **$** in the text box. In the displayed **Insert Variable** dialog box, select a variable type.

     If you set **Type** to **Custom**, set the following parameters:

     i. **Range**: variable range.

     ii. **Name**: name of a custom variable. Specifies the name of a variable when the variable is added.

     If you set **Type** to **System**, select a function. For details, see **Table 7-14**.

Table 7-14 Functions

Function	Description
ID card number	Randomly generates an ID number.
Mobile phone number	Randomly generates a phone number.
Random number in a range	Randomly generates an integer within your specified range.
Random character string	Randomly generates a string consisting of lowercase letters and numbers based on your specified string length (1 to 32 characters).
Time stamp	Generates a timestamp for the current time based on your specified unit. A 10-digit timestamp represents time in seconds whereas a 13-digit timestamp represents time in milliseconds.

Function	Description
Datetime Format	Outputs the current time in the specified format. Time offsets in seconds are supported.
UUID	Randomly generates a string of 32 characters.

If you set **Type** to **Four Fundamental rules**, select a function.

i. Select **+**, **-**, **x**, **/**, or **Scale Mode** from the drop-down list box, and click **Select**.

ii. Click . In the displayed **Four Functions Editing** dialog box, set the following parameters:

- **Expression**: Enter operands for the arithmetic operation. An operand can be a number (such as 1.23), a global, local, or system variable. You can reselect the operation here.
- **Mode**: Select how the expression result will be rounded. For details, see **Table 7-15**.

Table 7-15 Rounding modes

Name	Description (Using Two Decimal Place Rounding as an Example)
Far-zero Rounding	Rounds the number in a way that the result heads away from zero. For example, 1.234 is rounded to 1.24, and –1.234 is rounded to –1.24.
Near-zero Rounding	Rounds the number in a way that the result heads towards zero. For example, 1.234 is rounded to 1.23, and –1.234 is rounded to –1.23.
Round Up	Rounds the number towards positive infinity. Positive numbers will be rounded up, whereas the unwanted digits of negative numbers will be simply discarded. For example, 1.234 is rounded to 1.24, and –1.234 is rounded to –1.23.
Round Down	Rounds the number towards negative infinity. The unwanted digits of positive numbers will be simply discarded whereas negative numbers will be rounded down. For example, 1.234 is rounded to 1.23, and –1.234 is rounded to –1.24.

Name	Description (Using Two Decimal Place Rounding as an Example)
Rounding	If the first digit to be discarded is smaller than 5, the digit and all the digits that follow are simply discarded. Otherwise, the last retained digit is increased by 1 in value. For example, 1.234 is rounded to 1.23 and –1.235 is rounded to –1.24.
Rounded to six	If the first digit to be discarded is smaller than 6, the digit and all the digits that follow are simply discarded. Otherwise, the last retained digit is increased by 1 in value. For example, 1.235 is rounded to 1.23 and –1.236 is rounded to –1.24.
Bankers	Rounds the number to its nearest even integer. For example. 1.2350000 is rounded to 1.24, 1.2250000 is rounded to 1.22, and 1.2250001 is rounded to 1.23.
No Rounding	The number is not rounded. If this mode is selected, no further action is required.

- **Precision**: The maximum number of decimal places is 32.

iii. When the configuration is complete, click **OK**. If you specify multiple logics for a parameter, all the logics are combined as a character string and the string is assigned to the parameter as its value.

When the variable type is the encryption function, you need to select the encryption type and the position where the variable is inserted. The encryption types are as follows:

- MD5: Encrypts the val data using MD5 and returns the encrypted data string.
- MD5-BASE64: Performs MD5-BASE64 encoding on val data and returns the encoded data string.
- SHA-1: Encrypts the input data using SHA-1 and returns the encrypted data string.
- SHA-224: Encrypts the input data using SHA-224 and returns the encrypted data character string.
- SHA-256: Encrypts the input data using SHA-256 and returns the encrypted data character string.
- SHA-384: Encrypts the input data using SHA-384 and returns the encrypted data string.
- SHA-512: Encrypts the input data using SHA-512 and returns the encrypted data character string.

- BASE64 encoding: Encodes the input data using BASE64 and returns the encoded data string.
- BASE64 decoding: Performs BASE64 decoding on the input data and returns the decoded data character string.

- **Mode**
  - If you select **Repeated assignment** from the drop-down list box, a value is assigned each time the parameter value is needed in a thread.
  - If you select **Assigned once** from the drop-down list, the parameter value is assigned only once in a thread.

Step 7 Click **Save**.

----End

Adding a Cycle Controller

Step 1 In the case list on the left of the **Cases** tab, select the test case for which you want to add a cycle controller.

Step 2 On the **Case Script** tab, choose **Add More** > **Cycle Controller**.

Step 3 Set parameters.

Number of Cycles: Enter a value between 1 and 1,000,000.

Exit Conditions: If the request meets the condition, the cycle ends. Both expressions and comparisons support variables and character strings. Currently, conditions cannot be combined. If you need to use a set or multiple comparisons, set a regular expression.

Step 4 Click **Add More** to specify more details for the cycle. Up to 10 layers of steps can be nested.

Step 5 Click **Save**.

----End

Adding a Condition Controller

Step 1 In the case list on the left of the **Cases** tab, select the test case for which you want to add a condition controller.

Step 2 On the **Case Script** tab, choose **Add More** > **Condition Controller**.

Step 3 Set parameters.

Judgment Condition: If a request meets the condition, the request, transaction, or other logic controller defined by the condition controller is executed. Both expressions and comparisons support variables and character strings. Currently, conditions cannot be combined. If you need to use a set or multiple comparisons, set a regular expression.

Step 4 Click **Add More** to add requests, transactions, or other logic controllers that are executed after conditions are met. Up to 10 layers of steps can be nested.

Step 5 Click **Save**.

----End

Adding a Rendezvous Point

The purpose of a rendezvous point is to block concurrency until either X concurrent requests are blocked or the waiting time is reached, at which point all of them are released at a time. Therefore, a rendezvous point can create large instant loads at different points of the test case.

Step 1 In the case list on the left of the **Cases** tab, select the test case for which you want to add a rendezvous point.

Step 2 On the **Case Script** tab, choose **Add More** > **Rendezvous Point**.

Step 3 Set parameters.

Waiting Time (ms): waiting time at the rendezvous point.

Concurrent Number: When the number of users reaches this quantity, the users waiting at the rendezvous are released.

Step 4 Click **Save**.

----End

7.4.8 Adding PerfTest Concurrency Weight and Throughput Controllers

Adding a Concurrency Weight Controllers

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, select the test case for which you want to add a concurrency weight controller.

Step 5 On the **Case Script** tab, choose **Add More** > **Concurrency Weight Controller**.

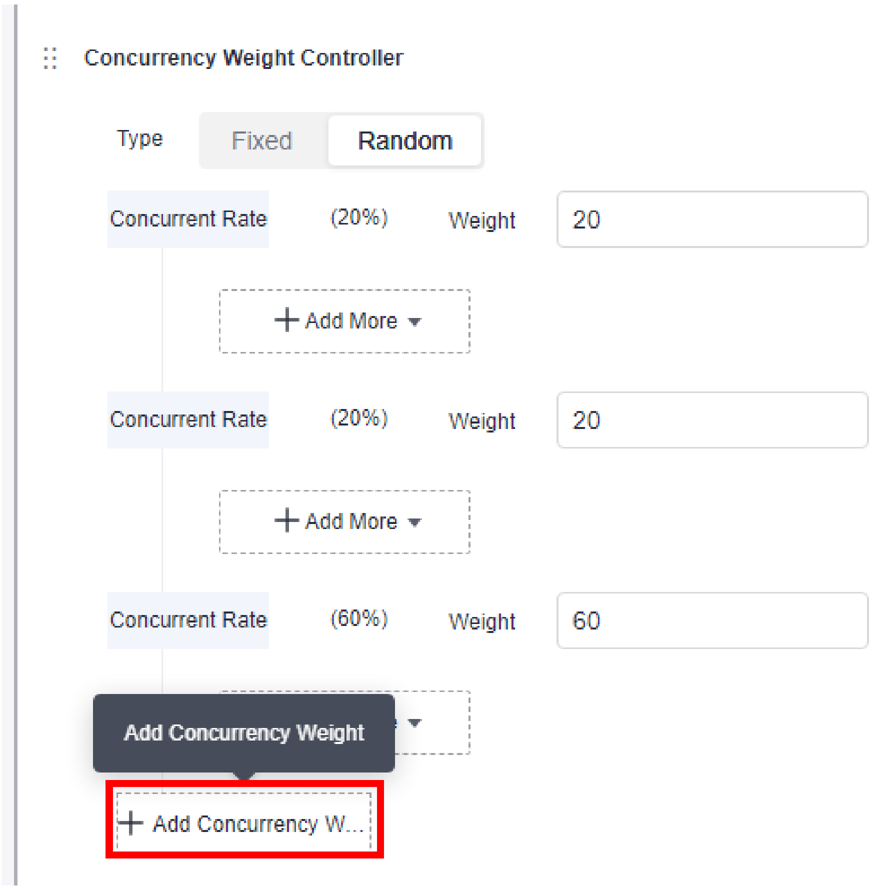
Step 6 Set parameters. Up to 10 layers of steps can be nested.

Type: The fixed branch and random branch modes are supported.

- **Fixed**: Randomly generates a fixed value for all concurrent executions. Subsequent executions use this fixed value to enter a fixed branch. This mode is used to group the concurrent users to simulate parallel tests in multiple scenarios in a case. Using multiple test cases for parallel testing in multiple low-concurrency scenarios requires many executors. You can group test scenarios in a case to save resources.
- **Random**: Generates a random value for each concurrent execution. Subsequent executions use this random value to enter a random branch. This mode is used to simulate the mixed ratio of the service interface RPS.

The concurrency weight is an integer greater than 0 and less than or equal to 10,000. If there is only one branch, the weight of the branch is 100%. You can click **Add Concurrency Weight** to add branches.

Figure 7-10 Concurrency weight controller



For example:

- **Fixed**: Assume that there are low-concurrency test scenarios A, B, and C, with 10, 20, and 30 concurrent users, respectively. The total number of concurrent users is 60. Create a test case with 60 concurrent users, set the concurrency weight controller to **Fixed**, and set the weight ratio to A:B:C = 1:2:3. In this way, the three scenarios are combined into one test case.
- **Random**: If the peak RPS of a service system is 1,000 (200 on interface A and 800 on interface B), perform the test by selecting **Random** and setting the weight ratio to A:B = 2:8. In this way, the final ratio of the RPS on interface A to that on interface B is 2:8.

Step 7 Click **Save**.

----End

Adding a Throughput Controller

Step 1 In the case list on the left of the **Cases** tab, select the test case for which you want to add a throughput controller.

Step 2 On the **Case Script** tab, choose **Add More** > **Throughput Controller**. Up to 10 layers of steps can be nested.

Step 3 Set parameters. Up to 10 layers of steps can be nested.

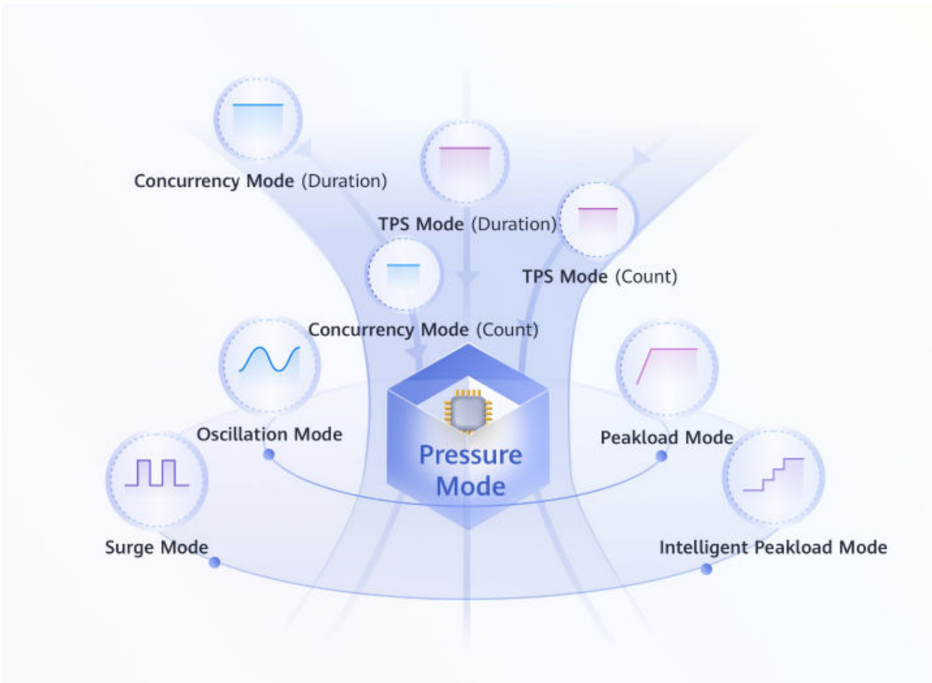
Set the RPS upper limit. The RPS executed by the controller cannot exceed the preset value. The RPS upper limit is **100** by default. The value can be up to 10 million and accurate to two decimal places.

Step 4 Click **Save**.

----End

7.4.9 Configuring a PerfTest Pressure Mode

Distilled from 30 years of high-concurrency test engineering solutions and practices, CodeArts PerfTest offers eight modes, such as surge (to simulate traffic bursts), intelligent peakload (to simulate system performance), oscillation (to simulate high and low peaks), and TPS mode (to customize pressure). You can create realistic scenarios quickly and increase the coverage of pressure test scenarios by 50%.



Configuring a Pressure Mode

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, click the test case you want to configure.

Step 5 Click **Pressure Configuration** and set phase parameters by referring to **Table 7-16**. A maximum of 1000 phases can be added to a case.

Table 7-16 Adding a phase

Parameter	Description
Phase Name	Phases names are used to distinguish different phases. For example, you can set the name to **Home page test**.
Pressure Mode	The following modes are supported: ● **Concurrency Mode**: The performance test is performed based on the number of concurrent users. The number of concurrent users can be a fixed number or can be set to ramp up throughout the phase. ● **TPS Mode**: The performance test is performed based on the number of transactions per second (TPS). In TPS mode, each execution of a test case by a virtual user is considered a transaction. The actual TPS may not reach your specified value because it will be affected by the specified maximum number of concurrent users and the average transaction latency. ● **Peakload Mode**: The pressure test is performed by increasing the number of concurrent users from the minimum to the maximum level over a specified ramp-up time, then maintaining that peak concurrency for the duration of the test. ● **Surge Mode**: This mode performs pressure testing using a fluctuating number of concurrent users. In each cycle, the test alternates between the minimum and maximum concurrency levels. The duration is based on the maximum concurrency and can be customized. The number of periods is the specified surge counts. ● **Oscillation Mode**: This mode performs pressure testing by periodically fluctuating the number of concurrent users. In each cycle, the load oscillates between the minimum and maximum concurrency levels. The number of periods is the specified oscillation counts. ● **Intelligent Peakload**: This mode performs a pressure test based on the peak concurrency achieved after the load increases gradually from a minimum concurrency to a maximum concurrency over a specified single-step execution duration. **NOTE** The executor performs a pressure test by gradually increasing the number of concurrent users based on the specified gradient. During the pressure test, the background generates phase data for each gradient. The system determines that the peak performance is reached when the data in a phase shows a clear performance inflection point compared with the data from the previous phase. The performance inflection point is identified in the following scenarios: The success rate falls below the threshold (100% by default) and the RPS curve shows a downward trend.

Parameter	Description
Execution Policy	This parameter is available only when the concurrency or TPS mode is used. The value can be **Duration** or **Count**. ● **Duration**: Tests will be performed based on the specified duration. ● **Count**: Tests will be performed based on the total number of tests.
Load Adjustment	This parameter is available only when the concurrency or TPS mode is used, and **Execution Policy** is set to **Duration**. Auto and manual load adjustments are supported. ● **Auto**: The maximum number of concurrent users of all stages is used as the maximum concurrency. ● **Manual**: The maximum number of concurrent users during a test is used as the maximum concurrency. After manual load adjustments, the original multi-stage pressure configuration becomes invalid.
When **Pressure Mode** is set to **Concurrency Mode**, **Execution Policy** is set to **Duration**, **Load Adjustment** is set to **Auto**, and **Gradient Increment** is disabled, set the following parameters.	
Concurrent Users	Concurrency refers to the number of users performing operations on a system at the same time. In CodeArts PerfTest, it is the number of virtual users you define when you configure a test phase.
Duration (min)	Duration that the phase will last.
When **Pressure Mode** is set to **Concurrency Mode**, **Execution Policy** is set to **Duration**, and **Load Adjustment** is set to **Manual**, set the following parameters.	
Maximum Concurrency	Maximum number of virtual users for a test case.
Start Concurrency	Starting number of virtual users for a test case.
Duration (min)	Duration that the phase will last.
When the pressure mode is set to **Concurrency Mode**, **Execution Policy** is set to **Duration**, **Load Adjustment** is set to **Auto**, and **Gradient Increment** is enabled, set the following parameters. Each case can have only one phase with **Gradient Increment** enabled.	
Start Concurrency	Starting number of concurrent users for a ramp-up.

Parameter	Description
Increasing Concurrency	Total number of concurrent users to be added. During a ramp-up, the starting number of concurrent users is simulated in the first subphase. The number of concurrent users in each subphase is the amount of (**Concurrency** in the previous phase + **Increasing Concurrency** x **Increment**). In the final subphase, the number of concurrent users will reach the amount of (**Start Concurrency** + **Increasing Concurrency**).
Duration per Pressure Test (min)	Duration that each subphase will last.
Increment	Number of concurrent users added in each subphase = **Increasing Concurrency** x **Increment**. The value can be **5%**, **10%**, **20%**, or **50%**.
When **Pressure Mode** is set to **Concurrency Mode** and **Execution Policy** is set to **Count**, set the following parameters. Only a single phase can be added.	
Concurrent Users	The number of concurrent users refers to the number of users performing operations on the system at the same time. In CodeArts PerfTest, it is the number of virtual users you define when you configure a test phase.
Total Sending Times	When the task is running, the performance test in this phase will be executed repeatedly until the number of execution times reaches the specified value. The total sending times must be greater than or equal to the number of concurrent users.
When **Pressure Mode** is set to **TPS Mode**, **Execution Policy** is set to **Duration**, and **Load Adjustment** is set to **Auto**, set the following parameters.	
Maximum Concurrency	Maximum number of virtual users for a test case.
TPS Value	Number of transactions executed per second.
Duration (min)	Duration that the phase will last.
When **Pressure Mode** is set to **TPS Mode**, **Execution Policy** is set to **Duration**, and **Load Adjustment** is set to **Manual**, set the following parameters.	
Maximum Concurrency	Maximum number of virtual users for a test case.
Start Concurrency	Starting number of virtual users for a test case.
TPS Value	Number of transactions executed per second.
Duration (min)	Duration that the phase will last.

Parameter	Description
When **Pressure Mode** is set to **TPS Mode** and **Execution Policy** is set to **Count**, set the following parameters. Only a single phase can be added.	
Maximum Concurrency	Maximum number of virtual users for a test case.
TPS Value	Number of transactions executed per second.
Total Sending Times	When the task is running, the performance test in this phase will be executed repeatedly until the number of execution times reaches the specified value. The total sending times must be greater than or equal to the number of concurrent users.
When **Pressure Mode** is set to **Peakload Mode**, set the following parameters.	
Start Concurrency	Starting number of virtual users for a test case.
Maximum Concurrency	Maximum number of virtual users for a test case.
Ramp Up(s)	Duration for which the number of virtual users increases from the start concurrency to the maximum concurrency. The recommended ramp-up time is shorter than the duration. If the ramp-up time is the same as the duration and the concurrency is large, the sampled maximum concurrency may be slightly less than the configured maximum concurrency.
Duration (min)	Duration that the phase will last.
Expected KPI	● Maximum response time: 60,000 ms ● Minimum success rate: 0 In peakload mode, the expected request response time of the tested system is less than or equal to the input value, and the request success rate is greater than or equal to the input value. If an indicator is unacceptable for six consecutive seconds for the first time (for example, the response time is greater than the expected value for six consecutive seconds), use the data in the first second as the analysis result. When the execution of a test case reaches the expected level, the task continues.
When **Pressure Mode** is set to **Surge Mode**, set the following parameters.	
Duration (min)	Duration that the phase will last.
Maximum Concurrency	Maximum number of virtual users for executing a test case, which is also the peak value during the changing number of virtual users.

Parameter	Description
Minimum Concurrency	Minimum number of virtual users for executing a test case, which is also the minimum value during the fluctuation of the number of virtual users.
Surge Count	Number of periods in which the number of virtual users fluctuates within the duration.
Peak Duration(s)	Duration of a stage based on the maximum number of concurrent virtual users in a period.
When **Pressure Mode** is set to **Oscillation Mode**, set the following parameters.	
Duration (min)	Duration that the phase will last.
Maximum Concurrency	Maximum number of virtual users for executing a test case, which is also the peak value during the fluctuation of the number of virtual users.
Minimum Concurrency	Minimum number of virtual users for executing a test case, which is also the trough value during the fluctuation of the number of virtual users.
Oscillation Count	Number of periods in which the number of virtual users fluctuates within the duration.
When **Pressure Mode** is set to **Intelligent Peakload**, set the following parameters.	
Start Concurrency	Starting number of virtual users for a test case.
Increase Concurrency	Number of virtual users increased by step. **Increase Concurrency** increases exponentially.
Single-step Execution Duration(s)	Duration of each incremental concurrency. You are advised to set this parameter to at least 20 seconds.
Duration (min)	Duration that the phase will last. You are advised to set this parameter to less than 30 minutes. When the success rate falls below the threshold (100% by default) and the RPS curve shows a downward trend, the pressure test stops.
Success Rate Threshold	The tolerable success rate. The default value is **100**.

Step 6 Click **Save**.

----End

7.4.10 Configuring Advanced PerfTest Options

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, click the test case you want to configure.

Step 5 Click **Advanced Configuration**.

1. **Executors**: The setting of the number of executors takes effect only when private resource groups are used. If this parameter is not set, the default policy is used. By default, concurrency supported by a single executor is 5,000 (HTTP/HTTPS), 5,000 (WebSocket), 5,000 (MQTT), 1,000 (JMeter), or 1,000 (HLS/RTMP/HTTP-FLV). Number of executors ≥ Maximum number of concurrent users in all phases/5,000.
2. **Log Collection Policy**: If there are too many requests in your case or cyclic nesting exists, use the request mode.
   - In **Request Mode**, the requests in a case generate logs separately.
   - In **Case Mode**, all requests in a case are associated with each other before logs are generated.

Step 6 Click **Save**.

----End

7.4.11 Configuring PerfTest SLAs

A Service Level Agreement (SLA) defines criteria for performance metrics. You can gain a better understanding of the performance of your service by checking if the metrics meet the criteria specified in the SLA.

Only PerfTest projects support SLAs. JMeter projects do not support. When the SLA metric thresholds are met, alarms will be generated or the performance tests will be stopped.

During a CodeArts PerfTest pressure test, the SLA sampling interval is 6 seconds. Any request that does not receive a response within this interval is not accounted for in the SLA metrics. If numerous requests experience high latency during the stress test, it may impact the SLA assessment.

You can create SLAs in either of the following ways:

- **Creating and Importing an SLA Template**. SLA templates can be imported for any PerfTest test tasks.
- **Create an SLA for a specific test case**. The SLA is applicable only to this test case.

Creating and Importing an SLA Template

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **SLAs** and click **Create SLA Template**. You can create up to 20 SLA templates in your account in each region.

Step 3 Set the SLA basic information.

- **SLA Name**: This is used to identify an SLA. The SLA name must be unique in a region.

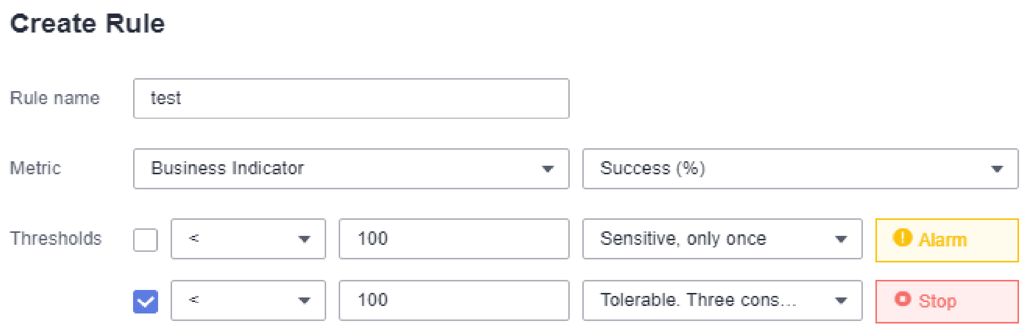
- **SLA Description**: This parameter is optional.

Step 4 Click **Add Rule**. On the **Create Rule** dialog box that is displayed, set the SLA rule by referring to **Table 7-17** and click **Confirm**.

Table 7-17 SLA rule parameters

Parameter	Description
Rule Name	This is used to identify a rule.
Metric	Three types of metrics are supported. CodeArts PerfTest will generate an alarm or stop a performance test when the metric reaches the specified threshold. • Average response time (RT) • Requests per second (RPS) • Success rate
Thresholds	Thresholds for triggering an alarm or stopping a performance test. For example, if you set the metric threshold for **Stop Test** to **< 100** and **3 consecutive times**, the test task will be stopped when the metric exceeds 100 for three consecutive times.

Figure 7-11 Creating SLA rules in a template



Step 5 (Optional) To create multiple rules, repeat **Step 4**. Up to five rules can be created for each SLA.

Step 6 Click **Save** to complete the SLA template creation. You can view the created SLA template on the **SLAs** page. To edit or delete an SLA template, locate the row that contains the SLA template, and click **Edit** or **Delete**.

Step 7 In the navigation pane, choose **PerfTest Projects**.

Step 8 Click the name of the desired project to go to the details page.

Step 9 In the **Case List** on the left, select the case for which you want to configure SLA rules.

Step 10 Click the **SLA Configuration** tab, click the rectangle on the right of **Import SLA Template**, select the name of the created SLA template from the drop-down list, and import the SLA template.

Step 11 In the **SLA Rules** area, select one or more requests to which the SLA rules will be applied.

You can also edit, delete, or add rules as required.

Step 12 Click **Save** to finish the SLA configuration.

----End

Creating SLA Rules for a Test Case

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, click the test case for which you want to configure SLA rules.

Step 5 Click the **SLA Configuration** tab.

Step 6 Click **Add Rule**. On the **Create Rule** dialog box that is displayed, set the SLA rule by referring to **Table 7-18** and click **Confirm**.

Table 7-18 SLA rule parameters

Parameter	Description
Rule Name	This is used to identify a rule.
Metric	Three types of metrics are supported. CodeArts PerfTest will generate an alarm or stop a performance test when the metric reaches the specified threshold. ● Average response time (RT) ● Requests per second (RPS) ● Success rate
Thresholds	Thresholds for triggering an alarm or stopping a performance test. For example, if you set the metric threshold for **Stop Test** to **< 100** and **3 consecutive times**, the test task will be stopped when the metric exceeds 100 for three consecutive times.

Step 7 In the **Target Requests** column, select one or more requests for each rule.

Step 8 (Optional) To create multiple rules, repeat 6 to 7. Up to five rules can be created for each SLA.

Step 9 Click **Save** to finish the SLA configuration.

----End

7.5 Configuring PerfTest Global Variables

7.5.1 Overview

Global variables are used to build data sets and enrich test data.

If a global variable is used in a request, variable values in the packet will be replaced with specified values during a pressure test.

Global variables can be used in many scenarios. For example, you can use different usernames and passwords to simulate a scenario with multiple users.

Global variables can be integer, text, or enumerated type. If there are a large number of global variables, for example, hundreds of user information records, you can use file variables to upload them in batches.

7.5.2 Adding PerfTest Global Variables

Adding a Global Variable of the Integer, Text, or Enumerated Type

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 On the **Cases** tab, click **Global Variables**.

Step 5 Click **Add Variable** and set parameters by referring to **Table 7-19**.

Table 7-19 Setting global variables

Parameter	Description
Name	Name of the global variable to be created. Enter 1 to 50 characters, including letters, digits, hyphens (-), underscores (_), and periods (.).
Type	Variable type, which can be integer, text, or enumerated.
Value	A pressure test task reads the value range of the corresponding global variable when it is running. For each virtual user, the variable value is polled. That is, the first virtual user obtains the first value of the variable, and the second virtual user obtains the second value. If the values are exhausted, the next virtual user obtains the first value again. Variable content is transmitted in plain text during the pressure test. Exercise caution when entering the content. ● When the variable type is **Integer**, enter the value range of the variable. ● When the variable type is **Enumerated**, click  to enter a variable value. ● When the variable type is **Text**, enter a text value.

Parameter	Description
Read Mode	**Sequential Mode**: Variables are read in sequence. **Random Mode**: Variables are read randomly.
Sharing Mode	**Case**: All concurrent requests share a variable. **Concurrency**: Each request copies a variable with the same name for use. Different concurrent requests read the variable values in sequence and do not affect each other.
Sensitive	If this option is selected, the variable value is encrypted for storage. Editing the variable after the page is refreshed will clear the variable value. Please exercise caution. Sensitive parameters apply only to text variables.

For details about the variable read rules for different variable read and sharing modes, see **Adding PerfTest Global Variables**.

Step 6 After the settings are complete, click **Save** in the **Operation** column to save the variables.

Step 7 After creating a variable, you can perform the following operations:

- Click **Edit** to modify it. If the global variable is referenced by a transaction, the value of the global variable in the transaction will be changed synchronously after the modification.
- Click **Delete** to delete it. Global variables that are referenced cannot be deleted.

----End

Adding a .csv or .xlsx Global Variable File

If a large number of parameters need to be dynamically obtained, you can use .csv or .xlsx files to dynamically transfer them during pressure tests.

Step 1 On the **Cases** tab, click **Global Variables**.

Step 2 Click **Download the .csv Template** or **Download the .xlsx Template** to obtain the required variable file template, and enter variables and values in the template. In the templates, enter variable names in the first row and enter variable values from the second row.

The restrictions on variable files are as follows:

1. Only .csv (UTF-8 without BOM) and .xlsx files can be imported.
2. The maximum length of a file name (including the extension) is 50 bytes.
3. Maximum file size:
   - .xlsx: 10 MB for the Professional edition and 20 MB for the Enterprise edition.
   - .csv: 10 MB for the Professional edition and 80 MB for the Enterprise edition by default.

Step 3 Click **Import File Variable** to import a .csv or .xlsx file.

- **Name**: name of the imported file
- **Type**: **File**
- **Value**: actual variable name. Separate multiple variable names by commas (,).
- **Read Mode**
  - **Sequential Mode**: Variables are read in sequence.
  - **Random Mode**: Variables are read randomly.
- **Sharing Mode**
  - **Case**: All concurrent requests share a variable.
  - **Concurrency**: Each request copies a variable with the same name for use. Different concurrent requests read the variable values in sequence and do not affect each other.

Step 4 After importing the global variables, you can perform the following operations:

- Click **Edit** to modify the variable read mode and sharing mode.

  For details about the variable read rules for different variable read and sharing modes, see **Adding PerfTest Global Variables**.
- Click **Download** to download the global variable file.
- Click **Delete** to delete the imported global variable file. Global variable files that are referenced cannot be deleted.
- If the imported global variable file is modified locally, click **Upload Again** to upload it again.

----End

Global Variable Read Rules

Table 7-20 lists the rules for reading manually specified or imported variables when different variable read modes and sharing modes are selected.

Table 7-20 Variable read rules

Variable Read Mode	Variable Sharing Mode	Description	Example
Sequential	Case	Each concurrency reads a variable value in sequence.	For example, for a task with a concurrency of 10, each concurrency reads a variable value in sequence.
Sequential	Concurrency	Each concurrency copies a variable with the same name for use. Different concurrent requests read the variable values in sequence and do not affect each other.	For example, for a task with a concurrency of 10, each concurrency reads the copied variable values in sequence.

Variable Read Mode	Variable Sharing Mode	Description	Example
Random	Case	Each concurrency randomly reads a variable value.	For example, for a task with a concurrency of 10, each concurrency randomly reads a variable value.
Random	Concurrency	Each concurrency copies a variable with the same name for use. Different concurrencies read the variable values randomly and do not affect each other.	For example, for a task with a concurrency of 10, each concurrency reads the copied variable values randomly.

Assume that the **number** variable has three values: **1**, **2**, and **3**, and there are two concurrencies: **A** and **B**.

- In sequential mode:

  Case mode: **A** reads **1** of **number**, and **B** reads **2** of **number**. The process goes on cyclically.

  Concurrency mode: **A** reads a copied variable **number** starting from **1**. **Concurrency mode**: **B** also reads a copied variable **number** starting from **1**.
- In random mode:

  The method of reading variable values is the same as that in sequential mode, regardless whether the **Case** or **Concurrency** sharing mode is used.

7.5.3 Inserting Variables in a PerfTest Request

When adding a request, enter **$** in the text box to insert a variable.

NOTE

Only global and local variables can be inserted into request names. Global variables can only be enumerated and text ones. A request name can contain up to 128 characters.

When Type is set to Custom, set the following parameters:

- **Range**: variable range.
- **Name**: name of a custom variable. Specifies the name of a variable when the variable is added.

When Type is set to System, select a function name.

- **ID card number**: Randomly generates an ID number.
- **Mobile phone number**: Randomly generates a phone number.
- **Random number in a range**: Randomly generates an integer within the specified range.

- **Random character string**: Randomly generates a string consisting of lowercase letters and numbers based on your specified string length (1 to 32 characters).
- **Time stamp**: Generates a timestamp for the current time based on your specified unit. A 10-digit timestamp represents time in seconds whereas a 13-digit timestamp represents time in milliseconds.
- **Time stamp(Format)**: The current time is output in the specified format. Time offsets in seconds are supported.

  The time format is described as follows:

  yyyy: indicates a four-digit year, for example, 2024.

  yy: indicates a two-digit year. For example, 24 indicates 2024.

  MM: indicates a two-digit month. For example, 01 indicates January.

  M: indicates a month without leading zeros. For example, 1 indicates January.

  dd: indicates a two-digit day, for example, 03.

  d: indicates a day without leading zeros, for example, 3.

  HH: indicates a 24-hour clock value. For example, 13 indicates 13:00.

  H: same as **HH**.

  hh: indicates a 12-hour clock value. For example, 01 indicates 01:00.

  h: indicates a 24-hour clock value without leading zeros. For example, 1 indicates 01:00.

  mm: indicates a minute, for example, 04.

  m: indicates a minute without leading zeros, for example, 4.

  ss: indicates a second, for example, 05.

  s: indicates a second without leading zeros, for example, 5.

  SSS: indicates a millisecond, for example, 098.
- **UUID**: randomly generates a string of 32 characters.

When the variable type is the encryption function, you need to select the encryption type and the position where the variable is inserted. The encryption types are as follows:

- MD5: Encrypts the val data using MD5 and returns the encrypted data string.
- MD5-BASE64: Performs MD5-BASE64 encoding on val data and returns the encoded data string.
- SHA-1: Encrypts the input data using SHA-1 and returns the encrypted data string.
- SHA-224: Encrypts the input data using SHA-224 and returns the encrypted data character string.
- SHA-256: Encrypts the input data using SHA-256 and returns the encrypted data character string.
- SHA-384: Encrypts the input data using SHA-384 and returns the encrypted data string.
- SHA-512: Encrypts the input data using SHA-512 and returns the encrypted data character string.
- BASE64 encoding: Encodes the input data using BASE64 and returns the encoded data string.

- BASE64 decoding: Performs BASE64 decoding on the input data and returns the decoded data character string.

7.6 Binding a Domain Name to a PerfTest Case

CodeArts PerfTest allows you to configure a Domain Name Service (DNS) address pool to map the domain name to the address. The domain name is defined in the URL and the IP address is automatically obtained through the DNS address pool.

- Domain names cannot be bound to MQTT transmission addresses.

Prerequisites

You can bind a domain name only when the **Request URL** on the **Packet** tab page contains the domain name.

Binding a Domain Name

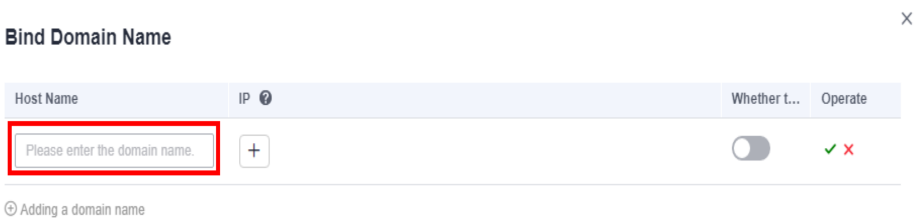
Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 On the **Cases** tab page, click **Domain name binding**

Step 5 Click **Adding a domain name**, and enter the domain name and IP address.



Step 6 Click the enable button, enter an IP address, and click .

----End

7.7 Configuring PerfTest Plug-ins

7.7.1 Overview

Plug-ins are available only to whitelisted users. To use this feature, service a **service ticket**.

CodeArts PerfTest allows you to use the Go language to develop custom requests, functions, and checkpoints.

- Custom requests can be used to implement interconnection of protocols such as database protocols, MQTT, and proprietary protocols.

- Custom functions can be used to assist in generating test data, such as generating specific encryption and decryption data and generating data using custom algorithms.
- Custom checkpoints are used for non-text, JSON, and other data serialization data or scenarios that are not supported by CodeArts PerfTest.

To implement custom requests, functions, and checkpoints, develop them according to **PerfTest Plug-in Development Specifications**, compile them into SO files according to **Guidelines for Compiling Go Plug-ins**, and use JSON files to describe them.

7.7.2 PerfTest Plug-in Development Specifications

Request Signatures

func(map[string]string) interface{}

Input parameters: CodeArts PerfTest built-in parameters and custom parameters.

Output parameters: The IResultV1 interface needs to be implemented. If necessary, use **IResultV1 Interface Implementation** provided by CodeArts PerfTest.

Table 7-21 CodeArts PerfTest built-in parameters

Name	Description	Remarks
__name	Name of a request	-
__goroutine_id	Goroutine ID	-
__executor_index	Executor index	The sequence number starts from 0.
__executor_count	Total number of executors	-

Function Signatures

func(map[string]string) string

Input parameters: custom parameters

Output parameters: character strings

Checkpoint Signatures

func(map[string]string) string

Input parameters: custom parameters

Output parameters: character strings. An empty character string indicates that the check is successful, and a non-empty character string indicates the check fails.

IResultV1 Interface Implementation

```
package main

import (
    "fmt"
    "time"
)

const (
    InnerVarName          = "__name"
    InnerVarGoroutineId   = "__goroutine_id"
    InnerVarExecutorIndex = "__executor_index"
    InnerVarExecutorCount = "__executor_count"
)

type IResultV1 interface {
    GetName() string
    GetUrl() string
    GetMethod() string
    GetRequestHeader() map[string]string
    GetRequestBody() string
    GetSentBytes() int
    GetResponseCode() int
    GetResponseHeader() map[string]string
    GetResponseBody() string
    GetReceivedBytes() int
    GetFailureMessage() string
    IsSuccess() bool
    GetBeginTime() int64
    GetEndTime() int64
    GetSubResults() []interface{}
}

//acquireResult generates root result.
//just call one time on the main func and generate sub result using parent.addSub()
func acquireResult(name string) *Result {
    result := &Result{}
    result.Name = name
    result.RequestHeader = map[string]string{}
    result.ResponseHeader = map[string]string{}
    result.ResponseCode = 200
    result.Success = true
    result.BeginTime = time.Now().UnixMilli()
    result.EndTime = time.Now().UnixMilli()
    return result
}

type Result struct {
    Name           string
    Url            string
    Method         string
    RequestHeader  map[string]string
    RequestBody    string
    SentBytes      int
    ResponseCode   int
    ResponseHeader map[string]string
    ResponseBody   string
    ReceivedBytes  int
    FailureMessage string
    Success        bool
    BeginTime      int64
    EndTime        int64
    SubResults     []interface{}
    SubIndex       int
}

func (r *Result) GetName() string {
    return r.Name
}
```

```
func (r *Result) GetUrl() string {
    return r.Url
}

func (r *Result) GetMethod() string {
    return r.Method
}

func (r *Result) GetRequestHeader() map[string]string {
    return r.RequestHeader
}

func (r *Result) GetRequestBody() string {
    return r.RequestBody
}

func (r *Result) GetSentBytes() int {
    return r.SentBytes
}

func (r *Result) GetResponseCode() int {
    return r.ResponseCode
}

func (r *Result) GetResponseHeader() map[string]string {
    return r.ResponseHeader
}

func (r *Result) GetResponseBody() string {
    return r.ResponseBody
}

func (r *Result) GetReceivedBytes() int {
    return r.ReceivedBytes
}

func (r *Result) GetFailureMessage() string {
    return r.FailureMessage
}

func (r *Result) IsSuccess() bool {
    return r.Success
}

func (r *Result) GetBeginTime() int64 {
    return r.BeginTime
}

func (r *Result) GetEndTime() int64 {
    return r.EndTime
}

func (r *Result) GetSubResults() []interface{} {
    return r.SubResults
}

//begin records begin time, do not forget call this function to update
func (r *Result) begin() {
    r.BeginTime = time.Now().UnixMilli()
}

//end records end time, do not forget call this function to update
func (r *Result) end() {
    r.EndTime = time.Now().UnixMilli()
}

//addSub adds sub result to parent, call this function adding sub result always.
//if name is not empty, renaming will be disabled
```

```
func (r *Result) addSub(name string) *Result {
    if name == "" {
        name = fmt.Sprintf("%s-%d", r.Name, r.SubIndex)
        r.SubIndex++
    } else {
        name = fmt.Sprintf("%s-%s", r.Name, name)
    }
    sub := acquireResult(name)
    r.SubResults = append(r.SubResults, sub)
    return sub
}
```

Table 7-22 Structure of the IResultV1 interface

Name	Description	Remarks
Name	Name of a request	The root result must be set using the built-in parameter **__name**.
Url	Request URL	-
Method	Method	Used for HTTP POST and GET.
RequestHeader	Request header	Used for HTTP
RequestBody	Request data	Do not record data with a large request size, for example, uploading a file. You only need to record the abstract.
SentBytes	Number of bytes sent	-
ResponseCode	Response code	Response status, which can be used for HTTP status codes or custom status codes. It is used to report the number of response statuses for analysis. Response code range: 100–599
ResponseHeader	Response header	Used for HTTP.
ResponseBody	Response data	Do not record data with a large response, for example, downloading a file. You only need to record the abstract.
ReceivedBytes	Bytes received	-
FailureMessage	Failure message	-
Success	Whether the request is successful	-

Name	Description	Remarks
BeginTime	Request start time	Unit: ms
EndTime	Request end time	Unit: ms
SubResults	Sub-request	When a custom request has multiple sub-requests, this field is used to record the execution status of each sub-request.
SubIndex	Sub-request index	When the addSub method of the parent result is used to generate a sub-result, and the sub-result name is not customized, this field is used to generate an index for generating the sub-result name.

7.7.3 Guidelines for Compiling Go Plug-ins

Constraint

The Go language has many restrictions on plug-ins, such as the Go version, third-party dependency version, and compilation parameters. The plug-ins must be strictly consistent with the main program of the CodeArts PerfTest executor. Otherwise, exceptions including but not limited to those described in **Table 7-23** may occur. Compile the **so** file strictly according to the conventions.

Table 7-23 Guidelines for compiling Go plug-ins

Item	Constraint Value	Description
arch	linux amd64	-

Item	Constraint Value	Description
Go version	Only Go 1.19.6 is supported.	Run the **go version** command to check the version. ● The following exception occurs when medium versions are inconsistent: fatal error: runtime: no plugin module data ● The following exception occurs when minor versions are inconsistent: plugin was built with a different version of package runtime
go.mod	github.com/go-resty/resty/v2 v2.7.0 github.com/shopspring/decimal v1.3.1 github.com/xuri/excelize/v2 v2.6.0 github.com/tidwall/gjson v1.14.4 github.com/valyala/fasthttp v1.43.0 go.uber.org/atomic v1.10.0 golang.org/x/net v0.8.0 golang.org/x/time v0.2.0	If a third-party library is directly or indirectly used, use the version according to the constraint value in the **go.mod** file. If the third-party versions are inconsistent, the following exception occurs: plugin was built with a different version of package github.com/json-iterator/go
TrimPath	Using TrimPath	If **TrimPath** is not used, the following exception occurs: plugin was built with a different version of package internal/unsafeheader Run the Linux command **strip** to remove symbols and debugging information.

Compilation Example

```
go build -buildmode=plugin -trimpath -o random.so random.go
strip random.so
```

7.7.4 JSON File Content Description

Example:

```
{
 "name": "random",
 "plugin": "random.so",
 "desc": "",
 "funcs": [{
        "name": "RandomNum",
        "type": "func",
        "desc": "",
        "params": [{
               "name": "min",
               "default": "0",
               "range": [],
               "desc": "this is a min",
               "required": true
        },
        {
               "name": "max",
               "default": "10",
               "range": [],
               "desc": "this is a max",
               "required": true
        }]
 }]
}
```

Table 7-24 JSON example description

Name	Description	Remarks
name	Key alias	Aliases are used for grouping.
plugin	Add-on name	Same as the name of the uploaded plug-in.
desc	Plug-in description	-
funcs	Function list	See **Table 7-25**.

Table 7-25 Function table

Name	Description	Remarks
name	Function name	Must be exportable.
desc	Function description	-
type	Function type	• **req**: request • **func**: function • **check**: checkpoint

Name	Description	Remarks
params	Parameters	See **Table 7-26**.

Table 7-26 Parameters

Name	Description	Remarks
name	Parameter name	Key to be used as an input parameter.
default	Default value	-
range	Value range	Example: ["on","off"]
desc	Description	-
required	Mandatory	-

7.7.5 PerfTest Plug-in Development Examples

This section provides PerfTest plug-in development examples, including:

- **Custom Request Example**
- **Example of a Custom Function**

Custom Request Example

Example of a custom Go request:

```
package main

import (
    "crypto/tls"
    "encoding/json"
    "errors"
    "fmt"
    "io/ioutil"
    "net/http"
    "net/http/cookiejar"
    "strings"
    "sync"
)

var (
    trs  = sync.Map{}
    jars = sync.Map{}
)

func getTr(addr string) *http.Transport {
    if v, ok := trs.Load(addr); ok {
        return v.(*http.Transport)
    } else {
        tr := &http.Transport{
            TLSClientConfig: &tls.Config{
                InsecureSkipVerify: true,
            },
            MaxConnsPerHost:     5000,
```

```
            MaxIdleConnsPerHost: 5000,
        }
        v, _ = trs.LoadOrStore(addr, tr)
        return v.(*http.Transport)
    }
}

func getJar(goroutineId string) *cookiejar.Jar {
    if v, ok := jars.Load(goroutineId); ok {
        return v.(*cookiejar.Jar)
    } else {
        jar, _ := cookiejar.New(nil)
        v, _ = jars.LoadOrStore(goroutineId, jar)
        return v.(*cookiejar.Jar)
    }
}

func outgoingLength(req *http.Request) int64 {
    if req.Body == nil || req.Body == http.NoBody {
        return 0
    }
    if req.ContentLength != 0 {
        return req.ContentLength
    }
    return -1
}

func redirectBehavior(reqMethod string, resp *http.Response, req *http.Request) (redirectMethod string,
shouldRedirect, includeBody bool) {
    switch resp.StatusCode {
    case 301, 302, 303:
        redirectMethod = reqMethod
        shouldRedirect = true
        includeBody = false

        if reqMethod != "GET" && reqMethod != "HEAD" {
            redirectMethod = "GET"
        }
    case 307, 308:
        redirectMethod = reqMethod
        shouldRedirect = true
        includeBody = true

        if resp.Header.Get("Location") == "" {
            shouldRedirect = false
            break
        }
        if req.GetBody == nil && outgoingLength(req) != 0 {
            shouldRedirect = false
        }
    }
    return redirectMethod, shouldRedirect, includeBody
}

func Do(params map[string]string) interface{} {
    parent := acquireResult(params[InnerVarName])
    sub := parent
    var (
        err           error
        redirectCount int
        req           *http.Request
        res           *http.Response
    )
    defer func() {
        if p := recover(); p != nil {
            sub.ResponseCode = 500
            sub.Success = false
            sub.FailureMessage = fmt.Sprintf("%v", sub)
        } else if err != nil {
```

```
            sub.ResponseCode = 500
            sub.Success = false
            sub.FailureMessage = err.Error()
        }
    }()

    redirect := params["redirect"]
    for {
        // process params
        url, method, body := params["url"], params["method"], params["body"]
        sub.Url, sub.Method, sub.RequestBody = url, method, body
        sub.SentBytes = len(body)

        // create request
        req, err = http.NewRequest(method, url, strings.NewReader(body))
        if err != nil {
            return parent
        }

        // load cookies
        jar := getJar(params[InnerVarGoroutineId])
        cookies := jar.Cookies(req.URL)
        for _, c := range cookies {
            req.AddCookie(c)
        }

        // write headers
        if params["headers"] != "" {
            headers := map[string][]string{}
            if err = json.Unmarshal([]byte(params["headers"]), &headers); err != nil {
                return parent
            }
            for k, vs := range headers {
                for _, v := range vs {
                    req.Header.Add(k, v)
                }
            }
        }

        for k, v := range req.Header {
            sub.RequestHeader[k] = strings.Join(v, ",")
        }

        // do request
        sub.begin()
        res, err = getTr(fmt.Sprintf("%s://%s", req.URL.Scheme, req.URL.Host)).RoundTrip(req)
        sub.end()
        if err != nil {
            return parent
        }
        var b []byte
        b, err = ioutil.ReadAll(res.Body)
        if err != nil {
            _ = res.Body.Close()
            return parent
        }

        // store cookies
        jar.SetCookies(req.URL, res.Cookies())

        for k, v := range res.Header {
            sub.ResponseHeader[k] = strings.Join(v, ",")
        }

        sub.ResponseBody = string(b)
        sub.ReceivedBytes = len(b)
        sub.ResponseCode = res.StatusCode
        _ = res.Body.Close()
        if redirect != "on" {
```

```
            break
        }
        if redirectMethod, shouldRedirect, includeBody := redirectBehavior(method, res, req); shouldRedirect {
            redirectCount++
            if redirectCount > 10 {
                err = errors.New("stopped after 10 redirects")
                return parent
            }
            params["url"] = res.Header.Get("Location")
            params["method"] = redirectMethod
            if !includeBody {
                params["body"] = ""
            }
            parent = acquireResult(params[InnerVarName])
            sub = parent
            continue
        }
        break
    }
    return parent
}
```

Example of a custom request in JSON format:

```
{
  "name": "Http 1.0",
  "plugin": "http.so",
  "desc": "Http plugin supports http 1.0 test.",
  "funcs": [
    {
      "name": "Do",
      "type": "req",
      "desc": "This is the entry of http request.",
      "params": [
        {
          "name": "url",
          "default": "",
          "range": [],
          "desc": "The url of http request",
          "required": true
        },
        {
          "name": "headers",
          "default": "{\"Content-Type\": [\"text/plain;utf-8\"]}",
          "range": [],
          "desc": "The headers of http request",
          "required": true
        },
        {
          "name": "method",
          "default": "GET",
          "range": [
            "GET",
            "HEAD",
            "POST",
            "PUT",
            "PATCH",
            "DELETE",
            "CONNECT",
            "OPTIONS",
            "TRACE"
          ],
          "desc": "The method of http request",
          "required": true
        },
        {
          "name": "body",
          "default": "",
          "range": [],
          "desc": "The body of http request",
```



```
          "required": true
        },
        {
          "name": "redirect",
          "default": "on",
          "range": [
            "on",
            "off"
          ],
          "desc": "If enable auto redirect",
          "required": true
        }
      ]
    }
  ]
}
```

Example of a Custom Function

Example of a custom Go function:

```
package main

import (
    "fmt"
    "math/rand"
    "strconv"
)

func RandomNum(params map[string]string) (s string) {
    var err error
    defer func() {
        if r := recover(); r != nil {
            s = fmt.Sprintf("%v", r)
        } else if err != nil {
            s = err.Error()
        }
    }()

    min, err := strconv.ParseInt(params["min"], 10, 64)
    if err != nil {
        return
    }
    max, err := strconv.ParseInt(params["max"], 10, 64)
    if err != nil {
        return
    }
    if min > max {
        min, max = max, min
    }
    return strconv.FormatInt(min+rand.Int63n(max-min+1), 10)
}

var (
    charSet = []rune("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789")
)

func RandomText(params map[string]string) (s string) {
    var err error
    defer func() {
        if r := recover(); r != nil {
            s = fmt.Sprintf("%v", r)
        } else if err != nil {
            s = err.Error()
        }
    }()

    length, err := strconv.ParseInt(params["length"], 10, 64)
    if err != nil {
```

```
        return
    }
    if length == 0 {
        return ""
    }
    set := []rune(params["charset"])
    if len(set) == 0 {
        set = charSet
    }
    n := len(set)
    rs := make([]rune, length)
    for i := int64(0); i < length; i++ {
        rs[i] = set[rand.Intn(n)]
    }
    return string(rs)
}
```

Example of a custom JSON function:

```
{
  "name": "Random",
  "plugin": "random.so",
  "desc": "Random plugin supports generating random value.",
  "funcs": [
    {
      "name": "RandomNum",
      "type": "func",
      "desc": "RandomNum generates random number.",
      "params": [
        {
          "name": "min",
          "default": "0",
          "range": [],
          "desc": "The maximum of random number.",
          "required": true
        },
        {
          "name": "max",
          "default": "100",
          "range": [],
          "desc": "The maximum of random number.",
          "required": true
        }
      ]
    },
    {
      "name": "RandomText",
      "type": "func",
      "desc": "RandomText generates random text of a specified length.",
      "params": [
        {
          "name": "length",
          "default": "32",
          "range": [],
          "desc": "The length of random text.",
          "required": true
        },
        {
          "name": "charset",
          "default": "",
          "range": [],
          "desc": "Random text is generated randomly from the specified character set. if empty, default char
set is 0-9 a-z A-Z",
          "required": false
        }
      ]
    }
  ]
}
```


7.7.6 Using a Plug-in in a PerfTest Case

After compiling the **so** and **JSON** files, upload the files to the PerfTest project and add custom requests and functions to the test cases.

Adding a Plug-in

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **PerfTest Projects**.

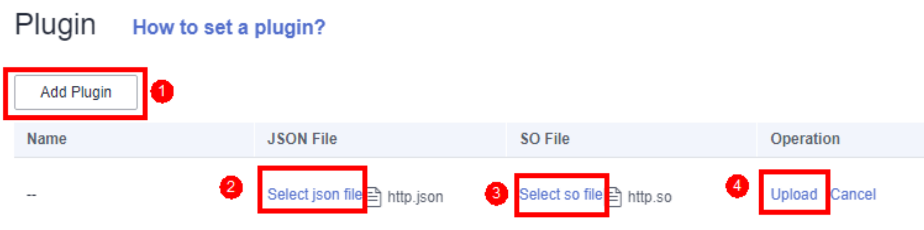
Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 On the **Cases** tab, click **Plugin**.

Step 5 In the displayed dialog box, click **Add Plug-in**, select the compiled JSON and SO files, and click **Upload**.

- Click **Select json file** and select the compiled JSON file.
- Click **Select so file** and select the compiled SO file.

Figure 7-12 Adding a plug-in



The file restrictions are as follows:

1. A **JSON** file cannot exceed 50 KB, and its name cannot exceed 64 characters.
2. Restrictions on **so** files: The file size cannot exceed 80 MB, and the number of files cannot exceed 20.

----End

Adding a Custom Request

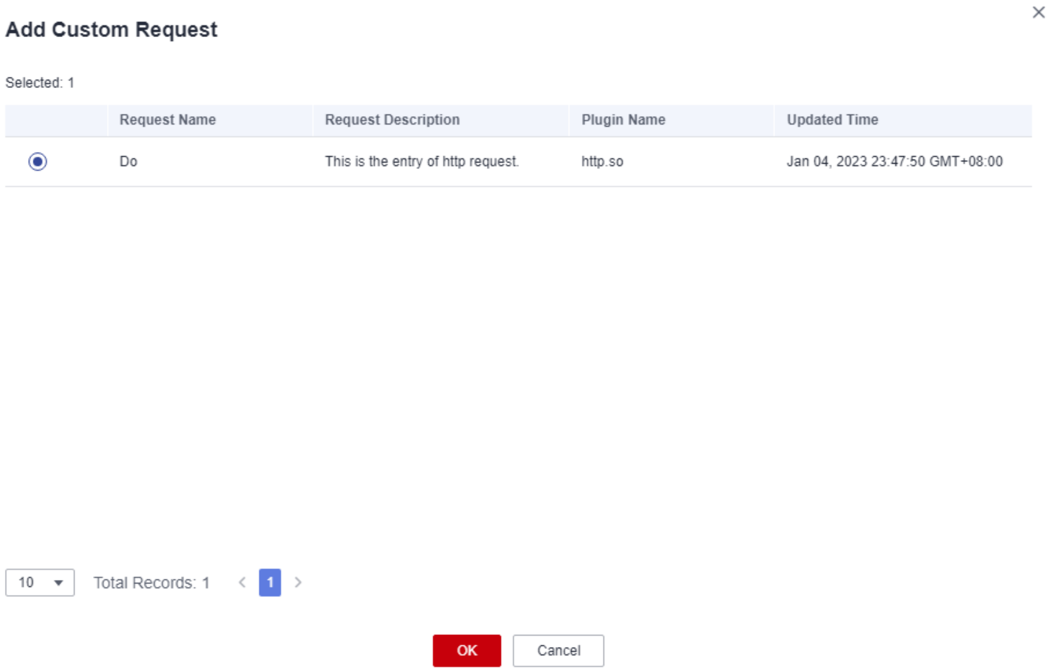
Ensure that you have added the custom request plug-in.

Step 1 In the case list on the left of the **Cases** tab, select the test case for which you want to add a custom request.

Step 2 On the **Case Script** tab, choose **Add Custom Request**.

Step 3 In the window that is displayed, select a request.

Figure 7-13 Adding a custom request



Step 4 Click **OK**.

----End

Adding a Custom Function

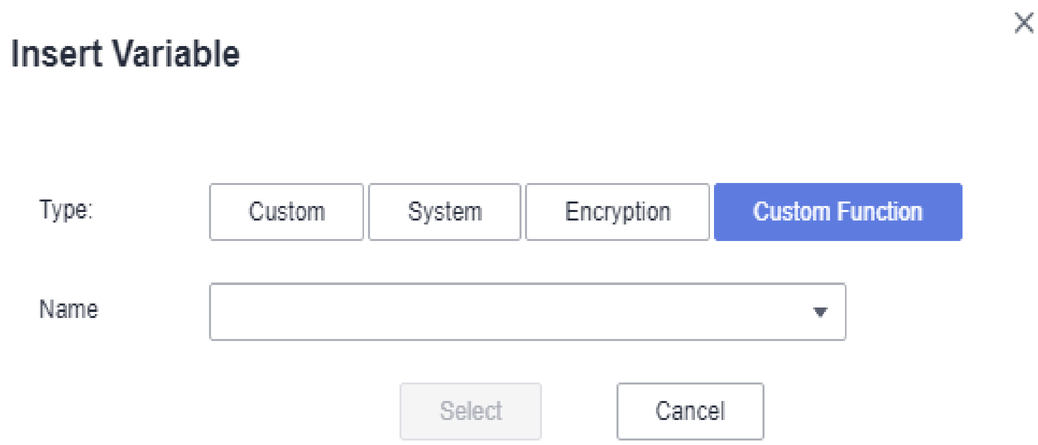
Before start, ensure that you have added the custom function plug-in.

Step 1 In the case list on the left of the **Cases** tab, select the test case for which you want to add a custom function.

Step 2 Enter **$** in the request content input boxes. The **Insert Variable** dialog box is displayed.

Step 3 Set **Variable Type** to **Custom Function** and select an imported function.

Figure 7-14 Adding a custom function



Step 4 Click **Select**.

----End

Adding a Custom Checkpoint

Ensure that you have added the custom checkpoint plug-in.

Step 1 In the case list on the left of the **Cases** tab, click the test case you want to edit.

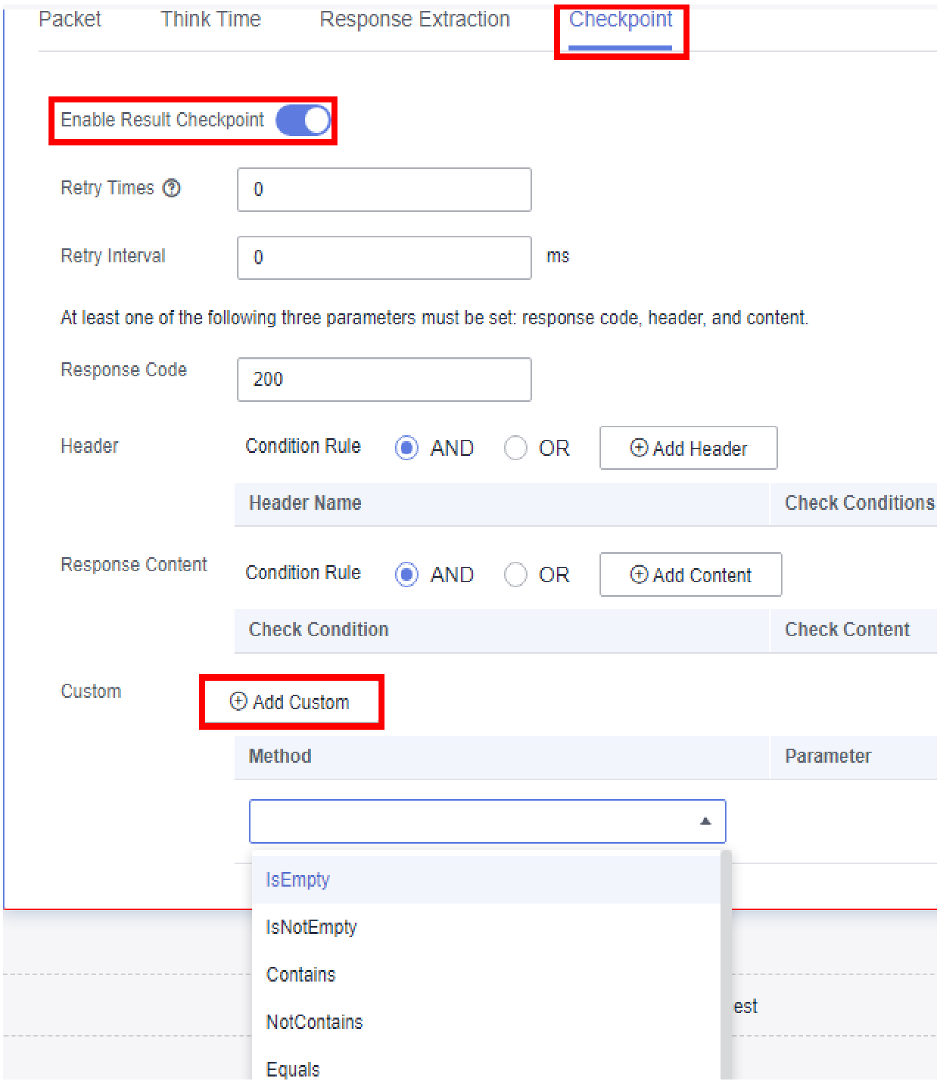
Step 2 On the **Case Script** tab, select the request for which a custom checkpoint is to be added and click in the operation column.

Step 3 On the **Checkpoint** tab, enable result check and set the response code, header, or response content.

When enabling the checkpoint, set at least one of the following three parameters: response code, header, and response content.

Step 4 Click **Add Custom**. In the displayed dialog box, select the imported checkpoints.

Figure 7-15 Adding a custom checkpoint



----End

7.8 Managing PerfTest Cases

Debugging a Test Case

After you create or edit a test case, you can debug it to detect syntax or configuration errors before pressure testing.

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 In the case list on the left of the **Cases** tab, click the test case you want to debug.

Step 5 Click **Debug** in the upper right corner of the page.

Step 6 Select the target test resource group as the executor and click **Start** to start debugging.

Step 7 Click **View log** to view the test case debugging details.

If an error is reported in the debugging result, edit the case based on the log information and debug it again.

Step 8 On the Debug log tab, view the debugging history.

----End

Resetting Configurations

When you modify a test case, you can reset the configurations that are not yet saved.

Step 1 In the case list on the left of the **Cases** tab, click the test case you want to configure.

Step 2 Click  in the upper right corner of the page. In the displayed dialog box, click **OK**.

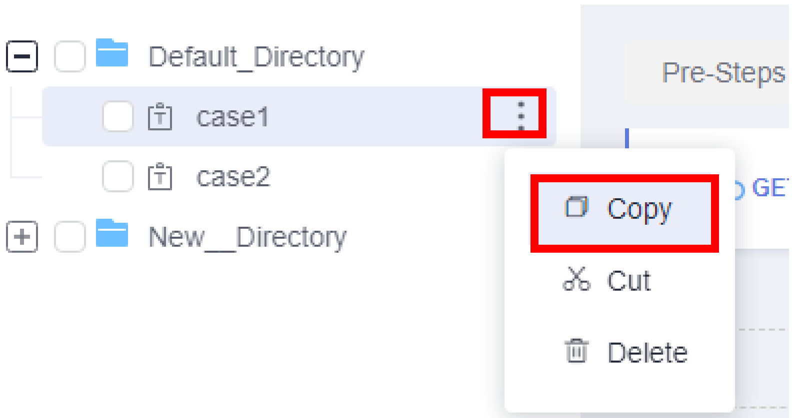
----End

Copying a Test Case

Step 1 In the case list on the left of the **Cases** tab, locate the test case you want to copy.

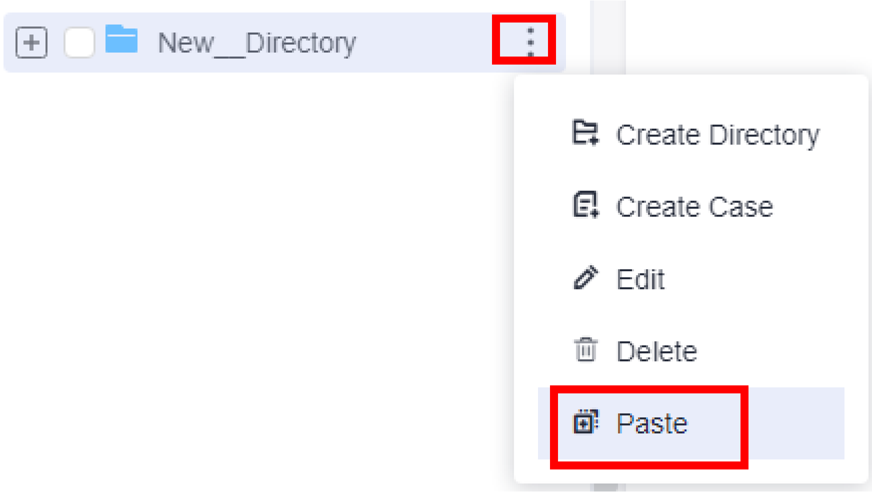
Step 2 Click next to the case, and choose **Copy** from the pop-up menu.

Figure 7-16 Copying a case



Step 3 Click next to the destination directory, and choose **Paste** from the pop-up menu.

Figure 7-17 Pasting a case



Step 4 Click **Save**.

----End

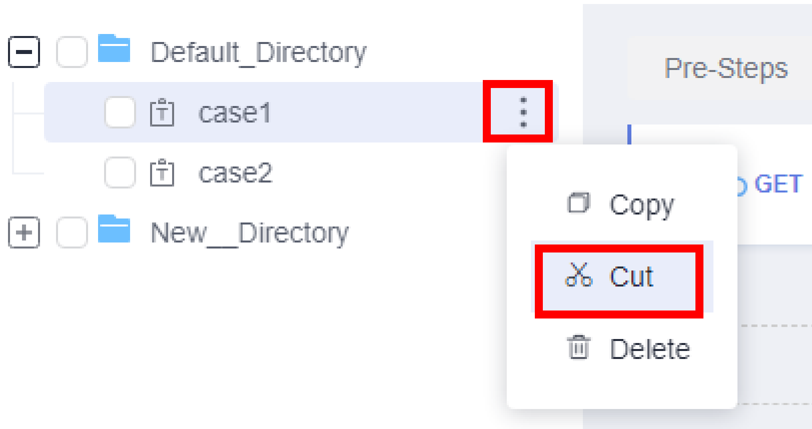
Moving a Test Case

Step 1 In the case list on the left of the **Cases** tab, locate the test case you want to move.

Step 2 Drag the case to move it to a new directory.

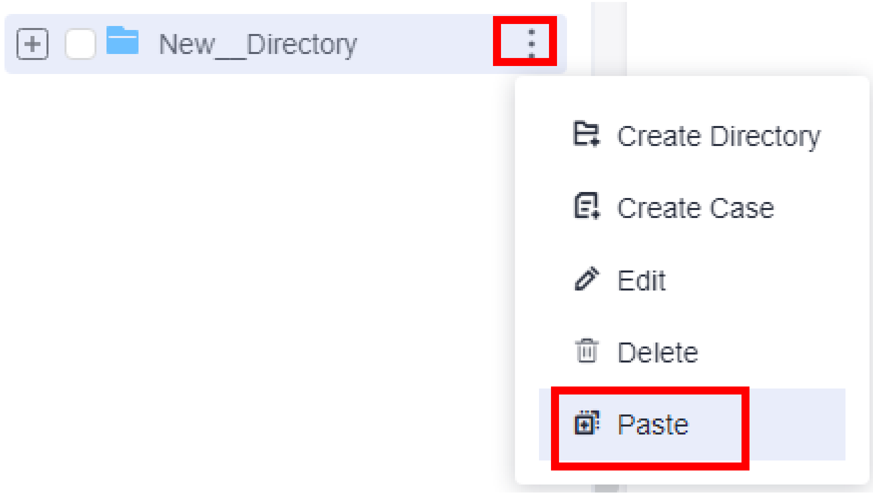
You can also click next to the case, and choose **Cut** from the pop-up menu.

Figure 7-18 Cutting a case



Click next to the destination directory, and choose **Paste** from the pop-up menu.

Figure 7-19 Pasting a case



----End

Deleting a Single Test Case

Step 1 In the case list on the left of the **Cases** tab, locate the test case you want to delete.

Step 2 Click next to the case, and choose **Delete** from the pop-up menu.

Step 3 Click **OK**.

----End

Deleting Test Cases in Batches

Step 1 In the case list on the left of the **Cases** tab, select multiple cases or directories.

Step 2 In the floating box displayed at the bottom of the page, click **Delete**.

Step 3 Click **OK**.

----End

Starting Test Cases in Batches

The case list supports batch operations, which apply only to test cases (including those inside a directory, but excluding the directory itself).

Step 1 In the case list on the left of the **Cases** tab, select multiple cases or directories.

Step 2 In the floating box displayed at the bottom of the page, click **Start**.

Step 3 In the displayed **Executing Temporary Test Task** dialog box, modify the parameters of the temporary task.

Table 7-27 Temporary test task parameters

Parameter	Description
Temporary Task Name	The current cases will not generate a permanent test task. The generated report will be displayed on the **Performance Reports** tab as a temporary task report and automatically deleted 30 days later.
Execution Mode	Select **Serial** or **Parallel**.
Resource Group Type	Select **Shared resource group** or **Private resource group**.

Step 4 Select **Confirm that this pressure test is approved and complies with local laws**.

Step 5 Click **Start** to execute the temporary task.

Step 6 After the test cases are started, click **View Report** to check the real-time performance report.

You can also switch to the **Performance Reports** tab in the upper left corner and click **Temporary Task List** to find the report by task name.

----End

8 PerfTest Task Management

8.1 Creating a PerfTest Task

A test task initiates a performance test based on a defined test model. It is a series of tests performed at different pressure points to continuously initiate pressure tests on a system. Testing allows you to obtain and analyze the performance data of the system. You can add multiple test tasks to a test project.

Prerequisites

- A test case has been created and configured based on service requirements.
- (Optional) A transaction has been added.

Creating a PerfTest Task

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 On the **Test Tasks** tab, click **Create Task**.

Step 5 Enter a task name and select an execution policy.

There are two execution policies:

- **Serial**: Test cases in the test task are executed in sequence.
- **Parallel**: Test cases in the test task are executed concurrently.

Step 6 Click **Add Case**. In the dialog box that is displayed, select created cases.

A test case can be added to multiple test tasks.

Step 7 Click **Save**.

----End

8.2 Starting a PerfTest Task

Prerequisites

- Test cases have been added.
- If using a private resource group, ensure that it is in the running state.
- If using a private resource group, ensure that ports 32001 and 32003 on the resource group's debugging node are enabled in the security group.
- Ensure network connectivity between the resource group's execution node and the applications to be tested.

Starting a PerfTest Task

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 Click the **Test Tasks** tab, and then click ▷ in the **Operation** column of the desired task.

Step 5 In the **Executing Test Task** dialog box, select a resource group type from the **Resource Group Type** drop-down list.

Step 6 Select **Confirm that this pressure test is approved and complies with local laws**.

Step 7 Click **Start** to start the test task.

Step 8 (Optional) After the test task is executed, click **View Report** or in the **Operation** column of the test task to view the real-time test report.

----End

8.3 Managing PerfTest Tasks

After a test task is created, you can manage it and the task phases.

Starting Test Tasks in Batches

Start multiple test tasks under the same test project.

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click the name of the desired PerfTest project to go to the details page.

Step 4 On the **Test Tasks** tab page, select multiple tasks and click **Start**.

Step 5 Select a resource group.

Step 6 Select **Confirm that this pressure test is approved and complies with local laws**.

Step 7 Click **Start**.

----End

Stopping Test Tasks

Step 1 On the **Test Tasks** tab, click next to the test task you want to stop.

Step 2 Select multiple tasks and click **Stop** to stop them at a time.

----End

Editing a Test Task

Step 1 On the **Test Tasks** tab, click in the **Operation** column of the test task you want to edit.

Step 2 Edit the test task name, execution policy, and cases as required.

- Click the original task name to display a text box. Enter a new task name in the box.
- Modify the execution policy.
- Delete or add test cases.

Step 3 Click **Save**.

----End

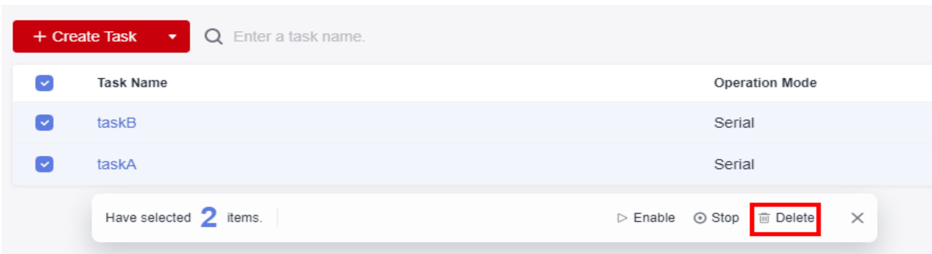
Deleting a Test Task

Deleted test tasks cannot be recovered. Exercise caution when deleting a task.

Step 1 On the **Test Tasks** tab, click in the **Operation** column of the test task you want to delete and choose **Delete**.

You can also select one or more test tasks and click **Delete**.

Figure 8-1 Deleting test tasks



Step 2 Delete the test tasks as prompted.

----End

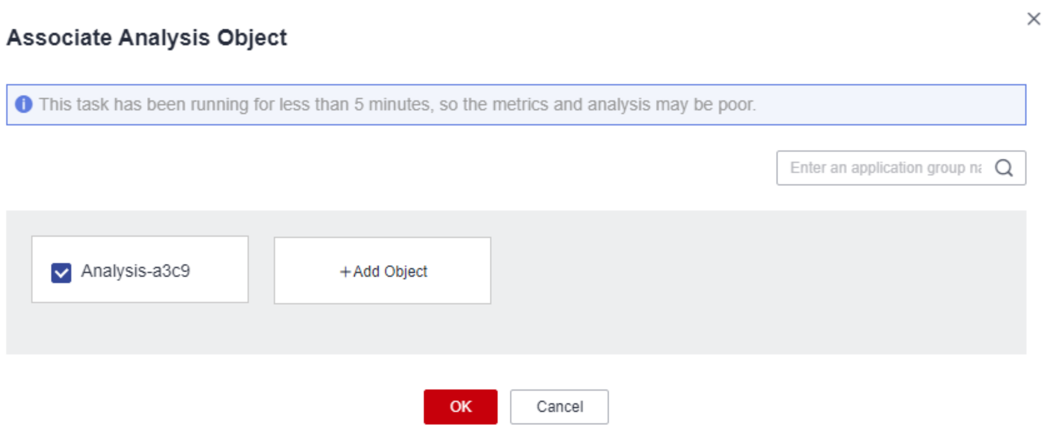
Associating Analysis Objects with a Test Task

This procedure describes how to connect to AOM and APM by associating analysis objects with a tested node. This allows you to monitor host resources and application tracings during a test. You can select and associate analysis objects with a test task before it is started. Once a test task has been started, analysis objects cannot be associated.

Step 1 On the **Test Tasks** tab, click in the **Operation** column of the test task to be associated with analysis objects, and then choose **Associate Analysis Object**.

Step 2 Select the analysis objects to be associated. To associate a new analysis object, click **Add Object** to create one by referring to **Configuring Intelligent Analysis for PerfTest Projects** and select the new object. Click **OK**.

Figure 8-2 Associating analysis objects



----End

9 PerfTest Report Management

9.1 Overview

CodeArts PerfTest delivers comprehensive pressure test reports that cover multiple dimensions and over 20 performance metrics, including transactions per second (TPS), response time (RT), success rate, TP*xx*, response code statistics, and execution logs. By integrating with real-time resource and tracing analysis based on visualized data, CodeArts PerfTest offers a comprehensive evaluation of performance metrics. It enables you to compare multiple rounds of reports, facilitating the rapid and accurate identification of performance bottlenecks and ensuring a smooth service rollout. CodeArts PerfTest provides both real-time and offline test reports, allowing you to view and analyze test data anytime.

Report Overview

This report shows the response performance of the tested system in a scenario with a large number of concurrent users. To understand the report, see the following information:

- **Statistical dimension**: In this report, RPS, response time, and concurrency are measured in a single case. If a request has multiple packets, the request is considered successful only when all the packets are responded to. The response time of the request is the sum of the response time of the packets.
- **Response timeout**: If the corresponding TCP connection does not return the response data within the set response timeout (5s by default), the case request is counted as a response timeout. Possible causes include: the tested server is busy or crashes, or the network bandwidth is fully occupied.
- **Verification failure**: The response packet content and response code returned from the server do not meet the expectation (the default expected response code of HTTP/HTTPS is 200), such as code 404 or 502. For details about the status codes, see **HTTP Status Codes**. A possible cause is that the tested service cannot be processed normally in scenarios with a large number of concurrent users. For example, a database bottleneck exists in the distributed system or the backend application returns an error.

- **Parsing failure**: All response packets are received, but some packets are lost. As a result, the entire case response is incomplete. This may be caused by network packet loss.
- **Bandwidth**: This report collects statistics on the bandwidth of the execution end of CodeArts PerfTest. The uplink indicates the traffic sent from CodeArts PerfTest, and the downlink indicates the received traffic. In the external pressure test scenario, you need to check whether the EIP bandwidth of the executor meets the uplink bandwidth requirement and whether the bandwidth exceeds the downlink bandwidth of 1 GB.
- **Requests per second (RPS)**: indicates the number of requests per second. Average RPS = Total number of requests in a statistical period/Statistical period.
- **SLA Result:** If you have configured SLA rules for a test case, you can view the SLA report to check whether the SLA rules are triggered during testing. In the SLA report, **Request Name** is the name of the request in the test case. **Rules** indicates the configured SLA rules. **Average Value** (displayed as the current value for a running task) indicates the average value of service metrics and the total number of times that SLA rules have been triggered since the pressure test starts. **Trigger Events** displays the request name and when and how many times the SLA rules were triggered during testing.
- **How to determine the quality of tested applications**: According to the service quality definition of an application, the optimal status is that there is no response or verification failure. If there is any response or verification failure, it must be within the defined service quality range. Generally, the value does not exceed 1%. The shorter the response time is, the better. User experience is considered good if the response time is within 2s, acceptable if it is within 5s and needs optimization when it is over 5s. TP90 and TP99 can objectively reflect the response time experienced by 90% to 99% of users.

For details about test reports, see **Table 9-1**.

Table 9-1 Test report description

Parameter	Description
Total Number of Metrics	Total number of metrics of all cases. ● **Maximum Concurrency**: the maximum number of concurrent virtual users. ● **RPS**: the number of requests per second. ● **TPS**: short for transactions per second. TPS = Number of transactions of a case/Running duration of the case. During a CodeArts PerfTest pressure test, all test steps in a case are executed cyclically. Each cycle is regarded as a transaction. ● **Response Time**: duration from the time when a client sends a request to the time when the client receives a response from the server. ● **Response Code**: distribution of response codes during the running of a pressure test task. ● **Bandwidth**: records the real-time bandwidth usage during the running of a pressure test task. ● **SLA Events**: occurrence of an event defined in the SLA. ● **Normal Response**: indicates the number of case responses that pass the set checkpoints. If no checkpoints are set, the number of case responses with code 2XX will be calculated by default. ● **Abnormal Response**: indicates the number of parsing failures, verification failures, response timeout, 3XX, 4XX, 5XX, and connection failures. ● **Success Rate**: Number of requests that are normally returned/Total number of requests.
Average RPS	Total number of requests in a statistical period/Statistical period
Average RT	Average response time of all requests sent in a second
Concurrent Users	Changes on the number of concurrent virtual users during testing
Bandwidth (kbit/s)	Records the real-time bandwidth usage during the running of a pressure test task. ● **Uplink bandwidth**: speed at which the CodeArts PerfTest execution node sends out data. ● **Downlink bandwidth**: speed at which the CodeArts PerfTest execution node receives data

Parameter	Description
Response distribution statistics	Number of cases processed per second for normal return, abnormal return, parsing failure, verification failure, response timeout, connection rejection, and other errors. This metric is related to the think time, number of concurrent users, and server response capability. For example, if the think time is 500 ms and the response time of the last request for the current user is less than 500 ms, the user requests twice per second. ● **Normal Response**: indicates the number of case responses that pass the set checkpoints. If no checkpoints are set, the number of case responses with code 2XX will be calculated by default. ● **Abnormal Response**: indicates the number of parsing failures, verification failures, response timeout, 3XX, 4XX, 5XX, and connection failures. ● **Parsing Failure**: indicates the number of HTTP responses that fail to be parsed. ● **Verification Failure**: indicates the number of case responses that do not pass the set checkpoints. If no checkpoints are set, the number of case responses returning contents other than 2XX will be calculated by default. ● **Response Timeout**: indicates the number of case requests that do not receive responses from the server within 5 seconds after a request packet is sent. ● **Rejected**: indicates the number of rejected connection requests. ● **Others**: indicates the number of other errors.
Response Code	**1XX/2XX/3XX/4XX/5XX** ● **1XX**: The request was received and understood and is being processed. ● **2XX**: The action requested by the client was received, understood, and accepted. ● **3XX**: The client must take additional actions, generally URL redirection, to complete the request. ● **4XX**: The request cannot be fulfilled due to an issue on the client side. ● **5XX**: The server failed to fulfill an apparently valid request.
Response Time Ratio	Ratio of response time of cases.

Parameter	Description
Maximum Response Time of TP (ms)	If you want to calculate the top percentile *XX* (TP*XX*) for a request, collect all the response time values for the request over a time period (such as 10s) and sort them in an ascending order. Remove the top (100–*XX*)% from the list, and the highest value left is the value of TP*XX*. Examples: ● TP50: Remove the top 50% from the list, and the highest value left is the value of TP50. ● TP75: Remove the top 25% from the list, and the highest value left is the value of TP75.

9.2 Viewing PerfTest Test Reports

You can view the monitored metrics during a performance test in a real-time test report. After a pressure test is complete, the system generates an offline test result report.

You can view test results by comparing reports generated for test tasks executed at different times or under different conditions.

Viewing PerfTest Test Reports

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Click next to the desired test project.

Step 4 On the **Performance Reports** tab, click a task name or in the **Operation** column to view the specific test report. For details about the parameters, see **Table 9-1**.

- If the test task is being executed, the real-time test report will be displayed. You can click **Stop Task** in the upper right corner of the page to stop the task.
- If the test task has been completed, the offline test report will be displayed. If the test task has been completed multiple times, multiple offline reports will be generated. In the **Report List** on the left, select the offline report you want to view. CodeArts PerfTest retains offline report data for one year. You can download offline reports (in PDF format) to your local PC and export the raw data (in CSV format) for further processing.

Step 5 To rename a test report, hover the cursor over the target name in the report list, click , and enter a new name.

Step 6 On the **Overview** tab, you can view the number of failed/total requests, average latency, RPS, maximum concurrency, success rate, SLA alarms, bandwidth, dynamic trend, and response codes.

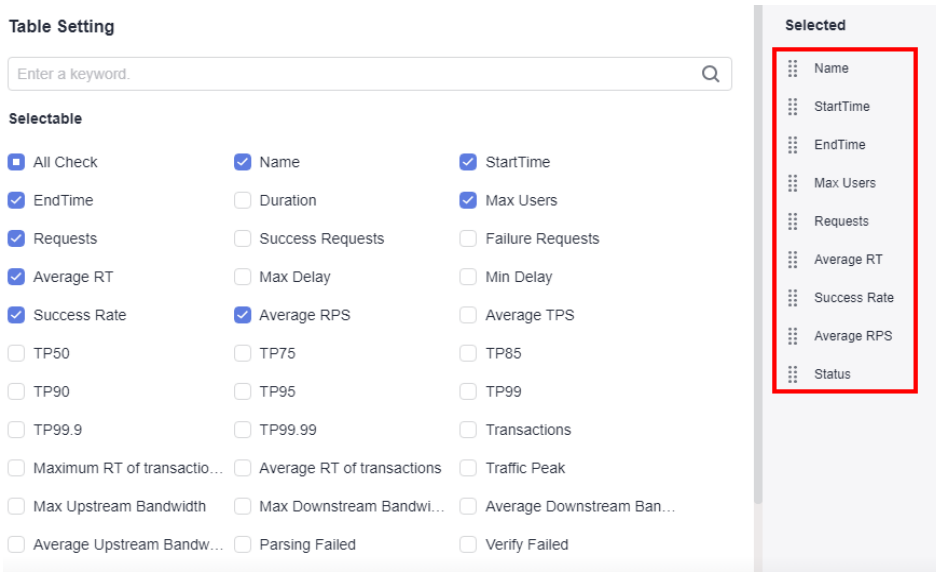
Step 7 On the **Detail** tab, you can view the logs, general test metrics, request details, and the SLA result of the test case.

1. Click **View Log**. In the displayed dialog box, view request logs, event logs, and pod information. Ten request logs are displayed based on the request name, return code, and result.

   Request logs whose case log collection policy is set to the request mode can be filtered by return code, result, or request name. The log collection policy is set in the advanced configuration of the test case. For details, see **Configuring Advanced PerfTest Options**.

2. During the execution of a test task, you can click **Process Voltage Adjustment** to dynamically adjust the pressure based on the real-time data. For the pressure mode, only the concurrency mode, TPS mode, and duration policy are supported.
3. The sampling modes displayed in the report details are equidistant sampling value and equidistant average value.
   - Equidistant sampling value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 30 minutes is displayed with sampling points at a fixed interval.
   - Equidistant average value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 30 minutes is displayed with average values at a fixed interval.
4. Click the **Data View** drop-down list box and enter a keyword to search for the required case or request data. You can also click the case or request data to be displayed in the case directory in the **Data View** drop-down list box.
5. On the **Detail** tab page, you can also click **List** to go to the report metric summary page.
   - Click **Customized Column**. In the displayed dialog box, select the items to be displayed, and drag the selected items in the **Selected** list on the right to change the item sequence.

     Figure 9-1 Table settings

   - Click  in the **Operation** column to view logs.

- Click in the **Operation** column to adjust the process voltage. For the pressure mode, only the concurrency mode, TPS mode, and duration policy are supported.
- Click in the **Operation** column to edit a test case.

Step 8 If the test task has been associated with **Application Monitor (APM)** in the intelligent analysis object, **Invoke Analysis** tab is displayed in the test report. On the **Invoke Analysis** tab, you can set search criteria to view the APM invoking details.

If the test task has been associated with **Node Monitor (AOM)** in the intelligent analysis object, **Monitored Indicator** tab is displayed in the test report. On the **Monitored Indicator** tab, you can view information such as CPU usage, memory usage, disk read speed, and disk write speed.

Step 9 View test results by comparing reports generated for test tasks executed at different times or under different conditions.

1. In the **Report List** area, click **Comparison**.
2. Select desired test reports and click **OK**.

   You can select a maximum of three offline reports for comparison. The first selected report is used as the baseline report.
3. Select a case from the **Test Case Comparison** box to compare the metrics of this case in different reports.
4. Test case comparison supports metric filtering.
   a. Click **Custom Comparison**. In the displayed dialog box, select the metrics to be displayed.
   b. (Optional) Drag the selected items in the **Selected** list on the right of the dialog box to change the item sequence.
   c. Click **OK**.

----End

10 PerfTest Transaction Management

10.1 Creating a PerfTest Transaction and Adding a Transaction Request

10.1.1 Overview

A transaction refers to a complete operation process from end to end, such as a login, criteria-based query, and payment. A transaction can be used by multiple test cases. CodeArts PerfTest supports flexible combination of transactions. You can create multiple transactions to a test project.

You can add a transaction request in any of the following ways:

- **Manually Adding Transaction Requests**
- **Adding Transaction Requests by Importing Recorded Scripts**
- **Adding Transaction Requests by Importing a Swagger File**

10.1.2 Manually Adding Transaction Requests

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Locate the desired PerfTest project and click to edit the transaction library.

Step 4 On the **Transaction Library** tab page, click **Create Transaction**, configure the following parameters, and click **OK**.

- **Transaction Name**: Enter a custom transaction name.
- **Type**: Normal transaction refers to transactions of performance tests in normal scenarios, including API tests of applications built based on HTTP/HTTPS/TCP/UDP/HLS/RTMP/WebSocket/HTTP-FLV/MQTT protocols. Transaction requests can be connected with each other.

Step 5 Click **Add Request** to add request information for the transaction. Then, click **Save**.

When the transaction type is **Normal**, you can add four types of request compositions: packet, think time, response extraction, and checkpoint. Packet is mandatory.

- **Packet**: data blocks transmitted between HTTP-based applications. For details, see **Adding PerfTest Request Information (Packet)**.
- **Think Time**: time interval between two actions. For details, see **Adding PerfTest Request Information (Think Time)**.
- **Response Extraction**: If multiple packets exist in the same transaction, use regular expressions or JSON to extract the output of the previous packet for the input of the next packet. For details, see **Adding PerfTest Request Information (Response Extraction)**.
- **Checkpoint**: used to verify if contents returned by servers are correct through customized verification information. For details, see **Adding PerfTest Request Information (Checkpoint)**.

Step 6 (Optional) A transaction contains one or more requests. To add more requests, click **Add Request**. Up to 40 requests can be added to a normal transaction.

----End

10.1.3 Adding Transaction Requests by Importing Recorded Scripts

In performance tests, virtual users simulate real users to use the tested system. The simulation process, such as **login, search, and purchase** in e-commerce applications, is implemented through recorded scripts. CodeArts PerfTest supports the record tool plug-in on the Chrome browser (version 26 or later versions). In the pressure test system, you can manually perform operations to capture request content and generate .json files. The recorded scripts simulate the actual operations of users, reducing the workload of manual packet writing and facilitating the use of the CodeArts PerfTest system.

Importing Recorded Scripts

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Locate the desired PerfTest project and click to edit the transaction library.

Step 4 On the **Transactions** tab, click **Import Script**. In the displayed dialog box, configure the following parameters:

- **Script Type**: Select **Manual recording script**.
- **Import to Transaction**: Select an existing transaction or create one. To create a transaction, click **Create Transaction**, enter a transaction name, and click .
- **Recorded Script File**: Click **Upload** and select a recorded script file. For details about the script recording plug-in and recording procedure, see **Preparing the Chrome Recording Plug-in** and **Recording Scripts**. The recorded script contains one or more packets.

Step 5 Click **Import**. A confirmation message displays upon successful script import.

Step 6 View the imported packets under the transaction. The number of packets successfully imported must match the number of requests in the uploaded script file.

Step 7 (Optional) Click to modify the imported scripts by referring to **Adding PerfTest Request Information (Think Time)** based on service requirements.

----End

Preparing the Chrome Recording Plug-in

CodeArts PerfTest's record tool requires Chrome 26 or later versions. Click in the upper right corner of the browser and choose **Help** > **About Google Chrome** to view the browser version. CodeArts PerfTest does not support recording operation scripts on the mobile WeChat.

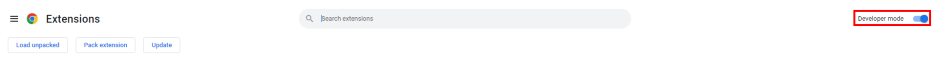
Step 1 **Download** the recording plug-in for Chrome.

Step 2 Decompress the installation package using a decompression tool.

Step 3 Enter **chrome://extensions/** in the address bar. The extension page is displayed.

Step 4 Enable the developer mode on the extension page.

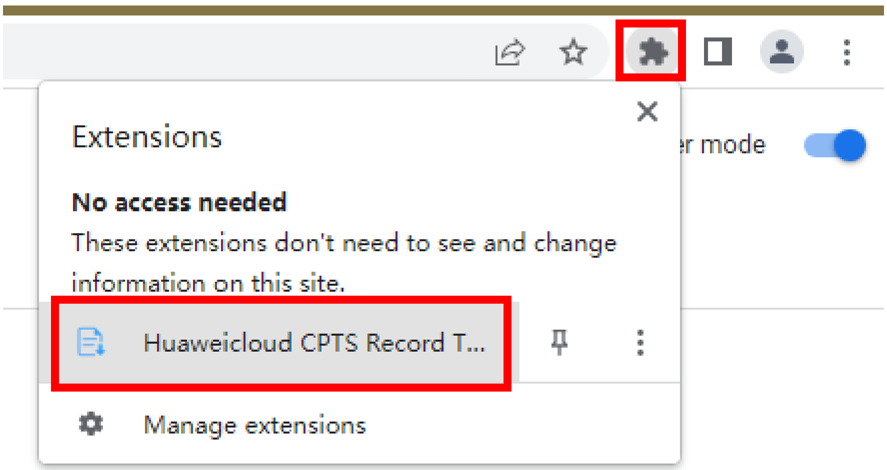
Figure 10-1 Enabling the developer mode



Step 5 Click **Load unpacked** and load the folder decompressed in **Step 2**.

Step 6 When the installation is complete, the icon of the recording tool is displayed in **Extensions** in the upper right corner of the browser.

Figure 10-2 Recording tool



----End

Recording Scripts

Use the CodeArts PerfTest recording tool to record scripts as follows:

Step 1 Open Chrome and click the icon of the CodeArts PerfTest recording tool in the upper right corner of the browser. The recording box is displayed.

Step 2 Click **Start Recording**.

Step 3 On the new tab page of the browser, enter the URL to be tested. The recording tool automatically records the HTTP requests during the access operation.

Step 4 When the recording is complete, switch to the recording tool page and click **Pause Recording**.

Step 5 By default, the record tool displays all types of requests. You can select the request composition based on the service requirements by clicking the **Filter Content** check box.

- other: others
- script: script
- xmlhttprequest: XML HTTP request
- main_frame: main frame
- stylesheet: style sheet
- image: image
- font: font file
- Ping: ping test

Step 6 If you select a request, the content of the request is displayed on the right of the page.

If the recorded content is not required or a new request needs to be recorded, perform the following steps to record scripts again:

1. Click **Clear Content** to clear the recorded request content.
2. Click **Start Recording**.
3. Perform operations on the new tab page and record again.

Step 7 Click **Export**, select the request to be exported, and click **Confirm Export**. The **Export Content** dialog box is displayed. Click **Generate .json File** to save the file to the local host. Plug-in recording script. The body type does not support the payload format.

Step 8 Close the tab where the record tool is located to disable the record tool.

----End

10.1.4 Adding Transaction Requests by Importing a Swagger File

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Locate the desired PerfTest project and click to edit the transaction library.

Step 4 On the **Transactions** tab page, click **Import Script**.

Step 5 Set **Script Type** to **Swagger file**, click **Upload**, and select the Swagger file to be uploaded. Only Swagger files in .yaml, .yml, and .json formats are supported.

Step 6 Click **Import**.

----End

10.2 Managing Transaction Requests

After a transaction request is created, you can insert, modify or delete it.

Inserting Request Information

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Locate the desired PerfTest project and click to edit the transaction library.

Step 4 On the **Transactions** tab, locate the desired transaction request, click in the upper right corner, and click **Insert a request**.

Step 5 Configure the request to be inserted and click **Save**.

----End

Modifying a Transaction Request

Step 1 In the left pane of the **Transactions**, click the transaction whose request information is to be modified.

Step 2 Click next to the transaction request to be modified to expand the request.

Step 3 Modify the transaction request as required and click **Save**.

----End

Disabling/Enabling a Transaction Request

If you want to retain a request of a transaction but do not want to use it, you can disable it. If you want to resume the use of the request, you can enable it again.

Step 1 Disable a transaction request.

On the **Transactions** tab page, select the transaction whose request is to be disabled, click next to the desired request to disable it. Other requests are not affected.

Step 2 Enable a transaction request.

On the **Transactions** tab page, select the transaction whose request is to be enabled, click next to the desired request to enable it. Other requests are not affected.

----End

Deleting a Transaction Request

Deleted transaction requests cannot be restored. Exercise caution when deleting a transaction request.

Step 1 On the **Transactions** tab, click the transaction whose request is to be deleted, and click next to the request.

Step 2 Delete the request as prompted.

----End

10.3 Managing PerfTest Transactions

After adding or modifying a transaction, you can quickly find syntax or configuration errors through debugging.

After a transaction is created, you can modify, delete, and copy the transaction.

A transaction used by a test cannot be deleted.

Debugging a Transaction

Prerequisites

- The resource group is in the **Running** state.
- Ports 32001 and 32002 on the debugging node of the resource group are enabled in the security group.
- The network between the debugging node of the resource group and the tested application is normal.

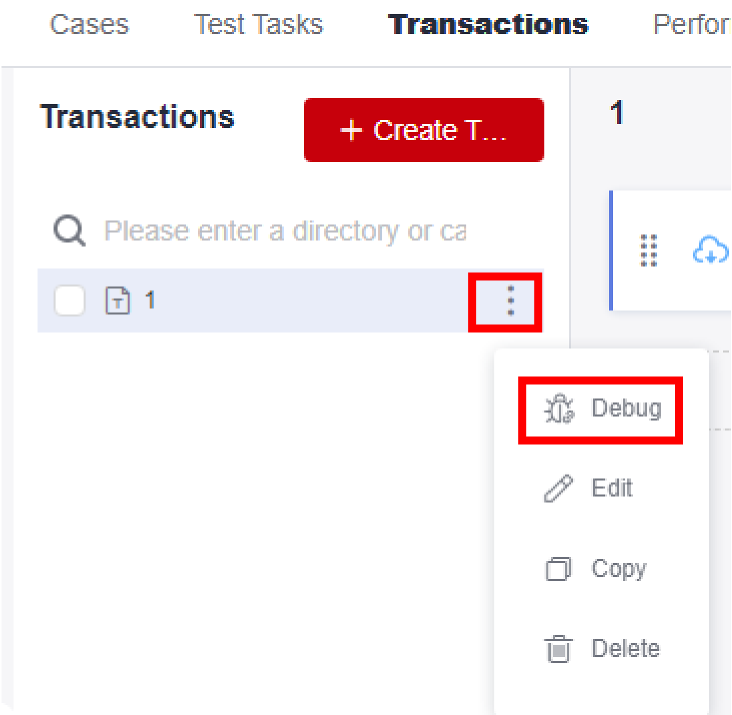
Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **PerfTest Projects**.

Step 3 Locate the desired PerfTest project and click to edit the transaction library.

Step 4 On the **Transactions** tab page, select the transaction to be debugged and click **Debug** in the upper right corner of the page. You can also click next to the transaction name and click **Debug**.

Figure 10-3 Debugging a transaction



Step 5 Select the target test resource group as the executor and click **Start** to start debugging.

Step 6 Click **View log** to view the transaction debugging details.

If an error is reported during the debugging, modify the transaction requests according to the log information, and then debug the transaction again.

Step 7 On the Debug log tab, view the debugging history.

----End

Modifying a Transaction

Step 1 On the **Transactions** tab, click next to the transaction name and choose **Edit**.

Step 2 Modify the transaction name and click **OK**.

----End

Deleting a Transaction

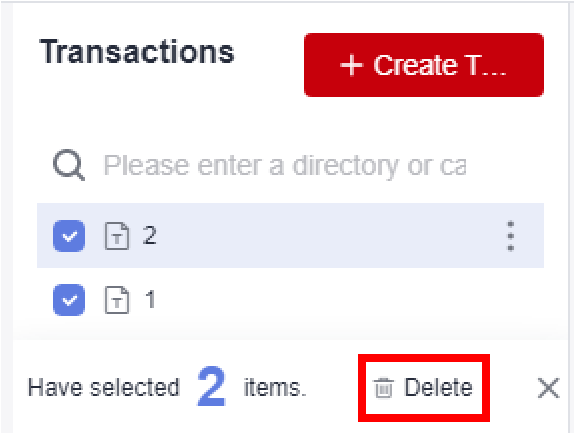
Deleted transactions cannot be restored. Exercise caution when deleting a transaction.

Step 1 On the **Transactions** tab, click next to the transaction name and click **Delete**. Delete the transactions as prompted.

You can also select a transaction name in the transaction list. In the floating box displayed in the lower part of the page, click **Delete** to delete the transaction as prompted.

If some transactions to be deleted in batches are referenced by a test task, only transactions that are not referenced can be deleted.

Figure 10-4 Deleting transactions



----End

Copying a Transaction

Step 1 On the **Transactions** tab, click next to the transaction name and click **Copy**.

Step 2 Enter the name of the new transaction and click **OK**.

----End

11 JMeter Test Project Management

11.1 Creating and Managing JMeter Projects

CodeArts PerfTest is compatible with JMeter scripts. JMeter test projects in CodeArts PerfTest support the native JMeter engine. You can import JMeter scripts into JMeter projects to quickly initiate high-concurrency performance tests with JMeter, and view complete performance test reports. Only private resource groups can be used for JMeter test projects.

Creating and Managing JMeter Projects

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **JMeter Projects**.

Step 3 Click **Create JMeter Project**.

Step 4 Configure the parameters by referring to **Table 11-1**.

Table 11-1 Creating a JMeter project

Parameter	Description
Project Name	Name of the new test project. The project name can contain a maximum of 128 bytes, including letters, digits, and the following special characters: _/.,|`'~;:-+=×...!@#$%^&*?()<>{}[]¥"
Description	Description of the new test project. Enter up to 50 characters.

Step 5 When the configuration is complete, click **OK**.

After the JMeter project is created, you can add a test plan for it. For details, see **JMeter Test Plan Management**.

Step 6 After the JMeter project is created, you can modify and delete it.

- In the JMeter project list, click in the row containing the desired project. Enter a new project name and click **OK**.
- In the JMeter project list, click > **Delete Project** in the row containing the desired project. Click **OK**.

----End

11.2 Configuring Intelligent Analysis for JMeter Projects

Intelligent analysis allows you to add analysis objects to your created JMeter test projects. Intelligent analysis is an enhanced capability. You can choose whether or not to configure it based on your requirements.

Currently, CodeArts PerfTest in the AP-Singapore region has been interconnected with APM 2.0, and the APM 1.0 configuration in CodeArts PerfTest is invalid. Use APM 2.0 when setting intelligent analysis.

Prerequisites

- APM has been enabled, a probe has been deployed, and an application has been generated. For details about how to deploy an APM probe, see **Monitoring Java Applications**.
- Application Operations Management (AOM) has been enabled, and a probe has been deployed. For details about AOM Agent deployment, see **Host Monitoring**.

Adding an Analysis Object

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **JMeter Projects**.

Step 3 In the test project list, click in the row containing the desired test project.

Step 4 On the **Intelligent Analysis** tab page, click **Add Analysis Object**.

Step 5 In the displayed dialog box, the system automatically generates the analysis object name, but you can change it. Select the analysis object to be added. The analysis object can be either an application group, that is **Application Monitor (APM)**, or a node, that is **Node Monitor (AOM)**. The application group is the monitoring group that is set when APM probes are installed on the application server. The node is the VM where AOM probes are installed.

Step 6 Click **OK** after the configuration.

Step 7 Edit or delete analysis objects.

- Edit an analysis object: On the **Intelligent Analysis** tab page, click next to the analysis object to modify and edit it as prompted. If an analysis object is associated with a test plan, once the object parameters are edited, those in the test plan are also changed.

- Delete an analysis object: On the **Intelligent Analysis** tab page, click next to the analysis object to be deleted and delete it as prompted. Analysis objects cannot be deleted if they are being used in a test plan.

----End

11.3 Configuring Resource Tags for a JMeter Project

You can add resource tags to created JMeter projects. Each JMeter project can be regarded as a resource. You can tag resources under your account for classification. In scenarios such as cost management, you can quickly view details about a type of resources by tag. Resource tags are an enhanced capability. You can choose whether or not to configure them based on your requirements.

Configuring Resource Tags

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **JMeter Projects**.

Step 3 In the JMeter test project list, click ··· > **Modify Tag** in the row containing the desired project.

Step 4 On the **Resource Tags** tab, click **Add Tag**. Select a tag key, and a new key-value pair text box is displayed. You can add multiple tags at a time.

- A test project can have a maximum of 20 tags. A tag key must be unique and can have only one value.
- Tag keys of different test projects can be invoked by each other. For example, add tag keys **a** and **b** to Test Project 1. When you add tags to Test Project 2, **a** and **b** will be automatically displayed in the tag key drop-down list.

Step 5 Click **OK** after the configuration.

Step 6 You can edit and delete tags.

- Editing a tag: On the **Resource Tags** tab, click next to the desired tag, enter another value, and click **OK**.
- Deleting a tag: On the **Resource Tags** tab, click next to the desired tag, and click **OK**.

----End

12 JMeter Test Plan Management

12.1 Creating a JMeter Test Plan

Overview

A JMeter test plan is a performance test activity based on the defined JMeter file, dependent JAR packages, and variable file. You can continuously initiate pressure tests on a system at different pressure test points to obtain and analyze its running performance data.

You can add multiple test plans to a test project.

The JMeter engine supports the following scripts:

- JMX scripts created using JMeter 5.2, 5.3, 5.4, or 5.6, without using any third-party plug-ins.
- Most scripts that use third-party plug-ins can be uploaded as JAR packages and that are implemented without modifying the ThreadGroup. However, CodeArts PerfTest does not guarantee the proper functioning of these scripts. You will need to debug them in CodeArts PerfTest.

The JMeter engine of CodeArts PerfTest does not support the following operations in scripts:

- Log output (Only requesting logs is supported.)
- Variables on the thread group configuration page

Creating a JMeter Test Plan

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **JMeter Projects**.

Step 3 Click in the row of the desired test project.

Step 4 On the **test plan** tab, click **Create Task Plan**.

Step 5 In the dialog box that is displayed, click **Select File**, select a .jmx file, and click **Upload**. When the file is imported, the dialog box closes automatically and the **test plan** tab is displayed. You can view the added test plan.

- The name (including the extension) of a JMX file can contain up to 64 bytes. The file size can be up to 10 MB.
- Basic JMeter scripts are supported, including thread group, HTTP request, HTTP Header Manager, request parameters, BeanShell, and timeout interval. You are advised to use a script of version 5.2.1 or later. Otherwise, a script parsing error may occur.

Step 6 On the **test plan** tab, click **Third Party Jar**. In the displayed dialog box, click **Select File**, select and import the JAR package on which the test plan depends. After the package is imported, close the dialog box.

When uploading a third-party JAR package, pay attention to the following:

- Ensure that the third-party JAR package is in the JMeter root directory **/lib/ext** when the local JMeter is working.
- Ensure that the script using the third-party JAR package can run properly on the local host.
- The name (including the extension) of a third-party JAR package can contain up to 64 bytes. The package size can be up to 80 MB.
- Java 9 and later versions do not support the upload of third-party JAR packages.

Step 7 View the newly added test plan on the **test plan** tab.

----End

Configuring a JMeter Pressure Mode

After creating a test plan, you can edit phase parameters.

Step 1 On the **test plan** tab, click the name of the test plan whose pressure is to be configured. On the **Plan Detail** page that is displayed, you can view the stages of different thread groups. Set the basic information by referring to **Table 12-1**.

Table 12-1 Pressure configuration

Parameter	Description
Execution Policy	The value can be **Duration** or **Count**. ● **Duration**: Tests will be performed based on the specified duration. ● **Count**: Tests will be performed based on the total number of tests.
Number of Threads	Number of concurrent virtual users.
Ramp-up Period	This parameter is involved only in **Phase1**. It corresponds to the ramp-up period in JMeter scripts. The parameter indicates the time required for increasing the number of concurrent users from 0 to the specified value.
Loop Count	This parameter is available only when **Execution Policy** is set to **Count**. Number of execution times of each virtual user.
Specify Thread Lifetime	This parameter is available only when **Execution Policy** is set to **Count**. If you enable this option, you can configure the **Duration** and **Startup Delay** parameters.
Duration	This parameter is available only when **Execution Policy** is set to **Count** and **Specify Thread Lifetime** is enabled. If the actual test duration reaches the maximum duration and the number of execution times does not reach the number of cycles, the thread group execution ends.
Duration	This parameter is available only when **Execution Policy** is set to **Duration**. How long the test will run. It is recommended that the duration be set to at least 300 seconds.
Startup Delay	When a test task is started, CodeArts PerfTest will wait the duration of startup delay before generating virtual users. This parameter can be configured when **Execution Policy** is set to **Count** and **Specify Thread Lifetime** is enabled. When **Execution Policy** is set to **Duration**, you can modify this parameter for **Phase1**.

Step 2 Click **OK**.

----End

Importing File Variables

If a large number of parameters need to be imported, you can directly import them using .csv or .xlsx files, enabling dynamic parameter transfer during pressure tests.

Step 1 On the **Plan Detail** tab, click **Variable File** in the upper right corner.

Step 2 In the displayed dialog box, click **Import File**.

Step 3 Select a .csv or .xlsx file and click **Upload**.

The restrictions on variable files are as follows:

1. Only .csv (UTF-8 without BOM) and .xlsx files can be imported.
2. The maximum length of a file name (including the extension) is 50 bytes.
3. Maximum file size:
   - .xlsx: 10 MB for the Professional edition and 20 MB for the Enterprise edition.
   - .csv: 10 MB for the Professional edition and 80 MB for the Enterprise edition by default.

----End

Configuring Global Variables for a JMeter Test Plan

Step 1 On the **Plan Detail** tab, click **Global Variable** in the upper right corner.

Step 2 In the displayed dialog box, click **Add Variable**.

Step 3 Enter a variable name, type, and value.

Variables are classified into static variables and evenly-split variables.

- Static variable: Content is delivered as a character string. When scripts are executed in distributed mode, the variable values obtained by each node are the same. For example, if the static variable "successRate = 0.8" is delivered, 2,000 tasks are concurrently executed by two actuators, and the value of **successRate** in the script of each actuator is **0.8**.
- Evenly-split variable: Content is delivered as an integer. When scripts are executed in distributed mode, the variable values obtained by each node are evenly distributed. Integer division is used during even distribution, and the remainder is allocated to one of the nodes. Examples:
  - If **tps**=**100** and there are four actuators, the value of the variable in the script of each actuator is 25.
  - If **tps**=**20** and there are three actuators, the value of the variable in the script of each actuator is 8, 6, 6.
  - If **tps**=**1** and there are four actuators, the value of the variable in the script of each actuator is 1, 0, 0, 0.

  Pay attention to the following when using the evenly-split variable:

  a. If the allocated value is sensitive to even split, set the number of actuators to ensure that the value of the variable is an integer multiple of the number of actuators.

b. If the allocated value is not sensitive to even split, increase the allocated value as much as possible to reduce the impact of the integer division on the remainder and evenly split the value.

c. If the allocated value cannot be 0, set the number of actuators to ensure that the value of the variable is greater than the number of actuators.

NOTE

If a variable configured in the global variable exists in **Test Plan** > **User Defined Variables** of a script, the variable value defined in the script will be overwritten.

Otherwise, the corresponding variable will be created in **Test Plan** > **User Defined Variables** of the script.

----End

Debugging a JMeter Test Plan

When a test plan is added or modified, you can debug it to quickly detect syntax or configuration errors and ensure that the model is available in a performance test.

Step 1 On the **test plan** tab, click the name of the task to be debugged. On the **Plan Detail** page that is displayed, click Debug in the upper part.

Step 2 On the debugging page, select a created private resource group as the executor and click **Start** to start debugging. You can view the debugging progress on the debugging page.

Step 3 If an error is reported in the debugging result, modify the .jmx file based on the error message, import the modified file, and debug the test plan again. For details, see **Editing a Test Plan**.

Step 4 Click **View log** to view the test plan debugging details.

Step 5 On the Debug log tab, view the debugging history.

----End

12.2 Starting a JMeter Test Plan

Prerequisites

- The resource group is in the **Running** state.
- Ensure that ports 32001 and 32003 on the resource group's debugging node are enabled in the security group.
- Ensure network connectivity between the resource group's execution node and the applications under test.

Starting a JMeter Test Plan

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **JMeter Projects**.

Step 3 Click in the row of the desired test project.

Step 4 Click in the row of the test plan you want to start.

Step 5 In the **Start Test Task** dialog box, select an enterprise project and a resource group. Only private resource groups are supported for executing JMeter tasks.

Step 6 Select **Confirm that this pressure test is approved and complies with local laws**.

Step 7 Click **Start** to start the test task. Click **View Report** to view the real-time performance report.

It is recommended that the duration of a pressure test be at least 300s and the number of concurrent users be set based on the actual situation. Obtain the maximum pressure value for the application through repeated tests by adjusting test data, and perform continuous optimization and verification for the application.

----End

12.3 Managing JMeter Test Plans

You can manage created test plans.

Starting Test Plans in Batches

Start multiple test plans in the same test project.

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **JMeter Projects**.

Step 3 Click in the row of the desired test project.

Step 4 On the **test plan** tab, select multiple test plans and click **Start**.

Step 5 Select a resource group, select **Confirm that this pressure test is approved and complies with local laws**, and click **Start**.

----End

Configuring Advanced Settings for a Test Plan

Step 1 On the **test plan** tab, click in the row containing the desired plan.

Step 2 Configure advanced parameters.

1. **Number of Executors**: This parameter takes effect only when private resource groups are used. If this parameter is not set, the default policy is used, that is, a single executor supports up to 1,000 concurrent users for JMeter. Number of executors ≥ Maximum number of concurrent users in all phases/1,000.
2. **Logs Not Collected**: By default, failed request logs are collected at a sampling rate of 10‰. This sampling rate can be adjusted to up to 1000‰.

Step 3 When the configuration is complete, click **OK**.

----End

Editing a Test Plan

Step 1 On the **test plan** tab, click in the **Operation** column of the desired plan, and choose **Update jmx**.

Step 2 In the displayed **Edit Test Plan** dialog box, import a new .jmx file, and then click **Close**.

Step 3 On the **test plan** tab, click in the **Operation** column of the desired plan, and choose **Variable File** to import the files to be referenced by the test plan.

You can also click a test plan name on the **test plan** tab to go to the test plan details page. On the displayed page, click **Variable File** to import the files to be referenced.

The file restrictions are as follows:

1. Only .csv (UTF-8 without BOM) and .xlsx files can be imported. When uploading a .csv file, ensure that it can be properly queried and used by JMeter on your local PC. Do not upload files in any other format, as this may lead to code reading errors.
2. The maximum length of a file name (including the extension) is 50 bytes.
3. The maximum size of an .xlsx file is 20 MB and that of a .csv file is 80 MB.

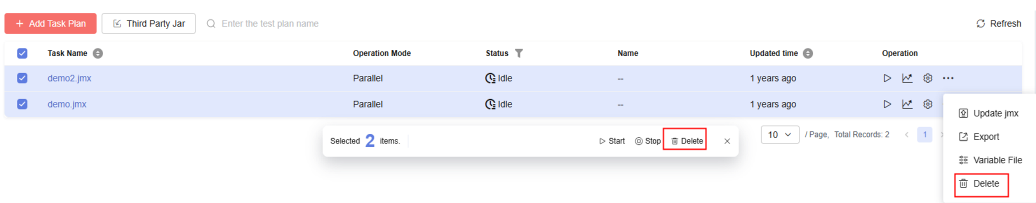
----End

Deleting a Test Plan

Deleted test plans cannot be recovered. Exercise caution when deleting a test plan.

Step 1 On the **test plan** tab, click in the **Operation** column of the desired plan, and choose **Delete**. You can also select multiple test plans and click **Delete** to delete them all.

Figure 12-1 Deleting one or more test plans



Step 2 Delete the test plans as prompted.

----End

Deleting Multiple Phases

Step 1 On the **test plan** tab, click the task name to go to the **Plan Detail** page.

Step 2 Select the desired thread group, click next to the phase to be deleted, and click **OK**.

Phase1 cannot be deleted.

----End

Exporting JMX Files, Variable Files, and JAR Packages

Prerequisites: The JMX files, variable files, and JAR packages have been imported.

Step 1 On the **test plan** tab, click in the **Operation** column of the desired plan, and choose **Export** to export its .jmx file.

Step 2 Click in the row containing the test plan whose variable file is to be exported and choose **Variable File**. The **Variable File** dialog box is displayed.

Step 3 Click **Export** in the row containing the variable file to be exported.

Step 4 Click **Third Party Jar**. The **Third Party JAR Package** dialog box is displayed.

Step 5 Click **Export** in the row containing the desired JAR package to export it.

----End

Associating Analysis Objects with a Test Plan

This procedure describes how to connect to AOM and APM by associating analysis objects with a tested node. This allows you to monitor host resources and application tracings during a test. You can select and associate analysis objects with a test plan before it is started. Once a test plan has been started, analysis objects cannot be associated.

Step 1 On the **test plan** tab, select the test task to be associated with analysis objects, and click .

Step 2 Select the analysis objects to be associated. To associate a new analysis object, click **Add Object** to create one by referring to **Configuring Intelligent Analysis for JMeter Projects** and select the new object. Click **OK**.

----End

13 JMeter Test Report Management

13.1 Overview

Real-time and offline JMeter test reports allow you to view and analyze test data at any time.

Report Overview

This report shows the response performance of the tested system in a scenario with a large number of concurrent users. To understand the report, see the following information:

- **Statistical dimension**: In this report, RPS, response time, and concurrency are measured in a single thread group. If a request has multiple packets, the request is considered successful only when all the packets are responded to. The response time of the request is the sum of the response time of the packets.
- **Response timeout**: If the corresponding TCP connection does not return the response data within the set response timeout (customized in *.jmx files), the thread group request is counted as a response timeout. Possible causes include: the tested server is busy or crashes, or the network bandwidth is fully occupied.
- **Verification failure**: The response packet content and response code returned from the server do not meet the expectation (the default expected response code of HTTP/HTTPS is 200), such as code 404 or 502. A possible cause is that the tested service cannot be processed normally in scenarios with a large number of concurrent users. For example, a database bottleneck exists in the distributed system or the backend application returns an error.
- **Parsing failure**: All response packets are received, but some packets are lost. As a result, the entire case response is incomplete. This may be caused by network packet loss.
- **Bandwidth**: This report collects statistics on the bandwidth of the execution end of CodeArts PerfTest. The uplink indicates the traffic sent from CodeArts PerfTest, and the downlink indicates the received traffic. In the external pressure test scenario, you need to check whether the EIP bandwidth of the

executor meets the uplink bandwidth requirement and whether the bandwidth exceeds the downlink bandwidth of 1 GB.

- **Requests per second (RPS)**: indicates the number of requests per second. Average RPS = Total number of requests in a statistical period/Statistical period.
- **How to determine the quality of tested applications**: According to the service quality definition of an application, the optimal status is that there is no response or verification failure. If there is any response or verification failure, it must be within the defined service quality range. Generally, the value does not exceed 1%. The shorter the response time is, the better. User experience is considered good if the response time is within 2s, acceptable if it is within 5s and needs optimization when it is over 5s. TP90 and TP99 can objectively reflect the response time experienced by 90% to 99% of users.

For details about the JMeter test report, see **Table 13-1**.

Table 13-1 Description of the JMeter test report

Parameter	Description
Total Number of Metrics	Total number of metrics in all thread groups. • **Maximum Concurrency**: indicates the maximum number of concurrent virtual users. • **RPS**: requests per second. • **Normal Response**: indicates the number of transaction responses that pass the set checkpoints. If no checkpoints are set, the number of transactions that return 2XX response code will be calculated by default. • **Response Time**: duration from the time when a client sends a request to the time when the client receives a response from the server. • **Response Code**: distribution of response codes during the running of a pressure test task. • **Bandwidth**: records the real-time bandwidth usage during the running of a pressure test task. • **Abnormal Response**: indicates the number of parsing failures, verification failures, response timeout, 3XX, 4XX, 5XX, and connection failures. • **Success Rate**: Number of requests that are normally returned/Total number of requests.
Average RPS	Total number of requests in a statistical period/Statistical period
Average RT	Changes of the average response time of a pressure test task
Concurrent Users	Changes on the number of concurrent virtual users during testing

Parameter	Description
Bandwidth (kbit/s)	Records the real-time bandwidth usage during the running of a pressure test task. ● **Uplink bandwidth**: speed at which the JMeter execution node sends out data. ● **Downlink bandwidth**: speed at which the JMeter execution node receives data.
Response distribution statistics	Number of transactions processed per second for normal return, abnormal return, parsing failure, verification failure, response timeout, connection rejection, and other errors. This metric is related to the think time, number of concurrent users, and server response capability. For example, if the think time is 500 ms and the response time of the last request for the current user is less than 500 ms, the user requests twice per second. ● **Normal Response**: indicates the number of transaction responses that pass the set checkpoints. If no checkpoints are set, the number of transactions that return 2XX response code will be calculated by default. ● **Parsing Failure**: indicates the number of HTTP responses that fail to be parsed. ● **Verification Failure**: indicates the number of transaction responses that do not pass the set checkpoints. If no checkpoints are set, the number of transactions that return 2XX response code will not be calculated. ● **Response Timeout**: number of case requests with no responses in the TCP connection within the configured response timeout interval. ● **Rejected**: indicates the number of rejected connection requests. ● **Others**: indicates the number of other errors.
Response Code	**1XX/2XX/3XX/4XX/5XX** ● **1XX**: The request was received and understood and is being processed. ● **2XX**: The action requested by the client was received, understood, and accepted. ● **3XX**: The client must take additional actions, generally URL redirection, to complete the request. ● **4XX**: The request cannot be fulfilled due to an issue on the client side. ● **5XX**: The server failed to fulfill an apparently valid request.
Response Time Ratio	Ratio of response time of cases.

Parameter	Description
Maximum Response Time of TP (ms)	If you want to calculate the top percentile *XX* (TP*XX*) for a request, collect all the response time values for the request over a time period (such as 10s) and sort them in an ascending order. Remove the top (100–*XX*)% from the list, and the highest value left is the value of TP*XX*. ● TP50: Remove the top 50% from the list, and the highest value left is the value of TP50. ● TP75: Remove the top 25% from the list, and the highest value left is the value of TP75.

13.2 Viewing JMeter Test Reports

View the monitoring data of each metric during a pressure test through the real-time test report. After a pressure test is complete, the system generates an offline test result report.

You can view test results by comparing reports generated for test plans executed at different times or under different conditions.

Viewing JMeter Test Reports

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **JMeter Projects**.

Step 3 Click next to the desired test project.

Step 4 On the **Performance Reports** tab, click a test plan name or in the **Operation** column to view the specific test report. For details about the parameters, see **Table 13-1**.

- If the test plan is being executed, the real-time test report will be displayed. You can click **Stop Task** to stop the executing test plan.
- If the test plan has been completed, the offline test report will be displayed. If the test plan has been completed multiple times, multiple offline reports will be generated. In the **Report List** on the left, select the offline report you want to view. CodeArts PerfTest retains offline report data for one year. You can download offline reports (in PDF format) to your local PC and export the raw data (in CSV format) for further processing.

Step 5 To rename a test report, hover the cursor over the target name in the report list, click , and enter a new name.

Step 6 On the **Overview** tab, you can view the number of failed/total requests, average RT, maximum concurrency, success rate, bandwidth, dynamic trend, and response codes.

Step 7 On the **Detail** tab, you can view the logs, common test metrics, and request details of the test plan.

1. Click **View Log**. In the displayed dialog box, view request logs, event logs, and pod information. Ten request logs are displayed based on the request name, return code, and result.

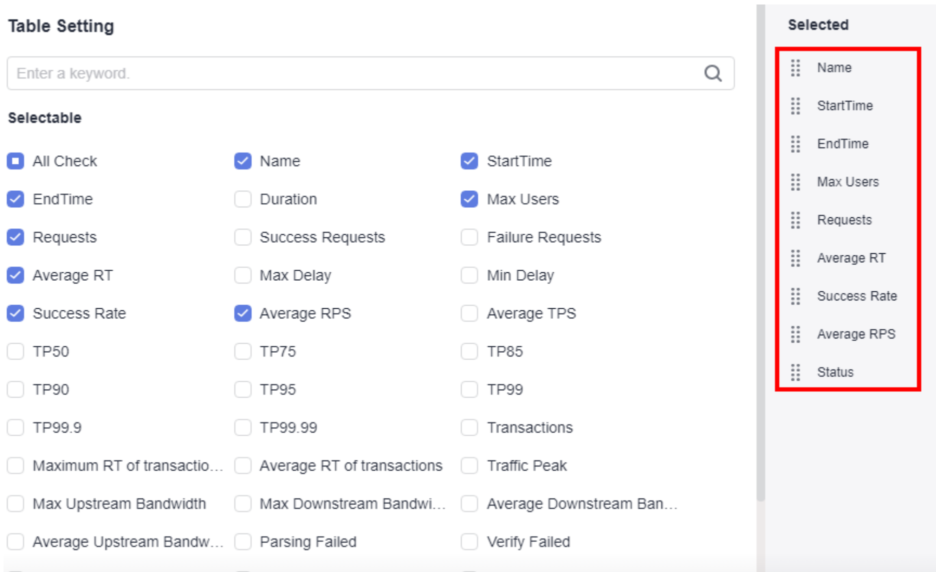
   Request logs can be filtered by return code, result, or request name.

2. The sampling modes displayed in the report details are equidistant sampling value and equidistant average value.
   - Equidistant sampling value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 30 minutes is displayed with sampling points at a fixed interval.
   - Equidistant average value: Based on the case execution duration, the trend chart of cases whose execution duration is longer than 30 minutes is displayed with average values at a fixed interval.

3. Click the **Data View** drop-down list box and enter a keyword to search for the required thread group or request data. You can also click the thread group or request data to be displayed in the directory in the **Data View** drop-down list box.

4. On the **Detail** tab page, you can also click **List** to go to the report metric summary page.
   - Click **Customized Column**. In the displayed dialog box, select the items to be displayed, and drag the selected items in the **Selected** list on the right to change the item sequence.

Figure 13-1 Table settings



   - Click  in the **Operation** column to view logs.
   - Click  in the **Operation** column to edit a thread group.

Step 8 If the test plan has been associated with **Application Monitor (APM)** in the intelligent analysis object, the **Invoke Analysis** tab is displayed in the test report. On the **Invoke Analysis** tab, you can set search criteria to view the APM invoking details.

If the test plan has been associated with **Node Monitor (AOM)** in the intelligent analysis object, the **Monitored Indicator** tab is displayed in the test report. On the **Monitored Indicator** tab, you can view information such as CPU usage, memory usage, disk read speed, and disk write speed.

Step 9 View test results by comparing reports generated for test plans executed at different times or under different conditions.

1. In the **Report List** area on the left, click **Comparison**.
2. Select desired test reports and click **OK**.

   You can select a maximum of three offline reports for comparison. The first selected report is used as the baseline report.
3. (Optional) Alternatively, click the **Comparison** tab in the upper part of the page. On the displayed **Comparison** tab, select the test reports to be compared.
4. Select a thread group from the **Thread Group Comparison** box to compare the metrics of this thread group in different reports.
5. Thread group metric comparison supports metric filtering.
   a. Click **Custom Comparison**. In the displayed dialog box, select the metrics to be displayed.
   b. (Optional) Drag the selected items in the **Selected** list on the right of the dialog box to change the item sequence.
   c. Click **OK**.

----End

14 Crontask Management

14.1 Creating a Crontask

Prerequisites

A PerfTest or JMeter project has been created, and test tasks are included.

Creating a Crontask

Step 1 Log in to the CodeArts PerfTest console.

Step 2 In the navigation pane, choose **Scheduled Test**.

Step 3 Click **Create**.

Step 4 Configure the parameters by referring to **Table 14-1**.

Table 14-1 Creating a crontask

Parameter	Description
Related Project	Select an existing PerfTest or JMeter project.
Related Task	Select the task for which you want to perform a scheduled pressure test. A task can be associated with only one scheduled pressure test task to be executed.
Crontask Name	Name of the crontask
Description	Description of the crontask

Parameter	Description
Resource Group	Select a test resource group for the crontask. ● You can select a shared resource group or an existing private resource group for a PerfTest project. Shared resource groups of a user can be bound to only one pressure test task. ● Only existing private resource groups can be selected for a JMeter test project.
Run Frequency	Frequency of running the crontask. Option: **Run only once** and **Run regularly**. ● **Run only once**: The task is executed only once at the time set in **Date of running**. ● **Run regularly**: The period settings can be **Monthly**, **Weekly**, and **Advanced**. – **Monthly**: The task is executed every month on the specified running date (one day or multiple days) and time. – **Weekly**: The task is executed every week on the specified running date (one day or multiple days) and time. – **Advanced**: The task is repeatedly executed based on crontab expressions. Use this mode if you are familiar with crontab expressions. The rules for entering running dates are as follows: Minute: integers ranging from 0 to 59, and operators * , - / Hour: integers ranging from 0 to 23, and operators * , - / Day (days in a month): integers ranging from 1 to 31, and the operator ? * , - / Month: Only ***** (monthly) is supported. Week (days in a week): any days in a week and the question mark (?) operator The day and week rules are mutually exclusive. When a non-default value is set for one parameter, the default value **?** is automatically set for the other parameter.
Deadline	Set this parameter only when **Run Frequency** is set to **Run regularly**. The task will not be executed after the deadline. The furthest deadline is one year later.

Step 5 When the configuration is complete, click **Sure**.

----End

14.2 Managing a Crontask

After a crontask is created, you can view, stop, and delete it.

Viewing a Crontask

You can view created crontasks.

Step 1 **Log in to the CodeArts PerfTest console**.

Step 2 In the navigation pane, choose **Scheduled Test**.

Step 3 You can view the overview of crontasks in the calendar on the left of the page. By default, crontasks for the day are displayed on the right of the page. Switch to the other tab to view all crontasks. They can be sorted by the next running time and filtered by the name or status.

----End

Stopping a Crontask

After a crontask is started at the configured running time, you can stop the test if the pressure test plan changes.

Step 1 On the **Crontask for the day** tab in the right pane of the **Scheduled Test** page, click the crontask name to be stopped.

Step 2 On the **Crontask Detail** page, click in the upper right corner.

Step 3 In the dialog box that is displayed, click **OK** to stop the crontask.

----End

Deleting a Crontask

Step 1 On the **Crontask for the day** or **All crontask** tab in the right pane of the **Scheduled Test** page, locate the crontask to be deleted.

Step 2 Click in the **Operation** column of the crontask to be deleted. In the displayed dialog box, click **OK** to delete it.

----End

15 CodeArts PerfTest Audit Trace Query

15.1 Key CodeArts PerfTest Operations Recorded by CTS

Cloud Trace Service (CTS) is a log audit service of the Huawei Cloud security solution. It records operations on cloud resources in your account, including collection, storage, and searching. You can use logs to perform security analysis, track resource changes, audit compliance, and locate faults.

Through CTS, you can record operations associated with Simple Message Notification (SMN) for later query, auditing, and backtrack operations.

Table 15-1 Key operations recorded by CTS

service_type	resource_type	resource_id	trace_name	Description
CPTS	cptsProject	Database primary key	createCptsProject	Creating a project
CPTS	cptsProject	Database primary key	deleteCptsProject	Deleting a project
CPTS	cptsProject	Database primary key	modifyCptsProject	Modifying a project
CPTS	cptsTask	Database primary key	createCptsTask	Creating a task
CPTS	cptsTask	Database primary key	deleteCptsTask	Deleting a task
CPTS	cptsTask	Database primary key	modifyCptsTask	Modifying a task
CPTS	cptsTemp	Database primary key	createCptsTemp	Creating a transaction

service_type	resource_type	resource_id	trace_name	Description
CPTS	cptsTemp	Database primary key	deleteCptsTemp	Deleting a transaction
CPTS	cptsTemp	Database primary key	modifyCptsTemp	Modifying a transaction
CPTS	cptsCase	Database primary key	createCptsCase	Creating a test case
CPTS	cptsCase	Database primary key	deleteCptsCase	Deleting a test case
CPTS	cptsCase	Database primary key	modifyCptsCase	Editing a test case
CPTS	cptsVariable	Database primary key	setVariable	Creating a variable
CPTS	cptsVariable	Database primary key	updateVariable	Modifying a variable
CPTS	cptsVariable	Database primary key	deleteVariable	Deleting a variable
CPTS	cptsTask	Database primary key	tempDebug	Debugging a task case
CPTS	cptsTaskStatus	Database primary key	updateTaskStatus	Updating task status

15.2 Viewing CTS Traces in the Trace List

Scenarios

Cloud Trace Service (CTS) records operations performed on cloud service resources. A record contains information such as the user who performed the operation, IP address, operation content, and returned response message. These records facilitate security auditing, issue tracking, and resource locating. They also help you plan and use resources, and identify high-risk or non-compliant operations.

What Is a Trace?

A trace is an operation log for a cloud service resource, tracked and stored by CTS. Traces record operations such as adding, modifying, or deleting cloud service resources. You can view them to identify who performed operations and when for detailed tracking.

What Is a Management Tracker and Data Tracker?

A management tracker identifies and associates with all your cloud services, recording all user operations. It records management traces, which are operations performed by users on cloud service resources, such as their creation, modification, and deletion.

A data tracker records details of user operations on data in OBS buckets. It records data traces reported by OBS, detailing user operations on data in OBS buckets, including uploads and downloads.

Constraints

- Before the organization function is enabled, you can query the traces of a single account on the CTS console. After the organization function is enabled, you can only view multi-account traces on the **Trace List** page of each account, or in the OBS bucket or the **CTS/system** log stream configured for the management tracker with the organization function enabled. For details about organization trackers, see **Organization Trackers**.
- You can only query operation records of the last seven days on the CTS console. They are automatically deleted upon expiration and cannot be manually deleted. To store them for longer than seven days, configure transfer to Object Storage Service (OBS) or Log Tank Service (LTS) so that you can view them in the OBS buckets or LTS log streams.
- After creating, modifying, or deleting a cloud service resource, you can query management traces on the CTS console 1 minute later and query data traces 5 minutes later.

Prerequisites

1. **Register with Huawei Cloud and complete real-name authentication.**

   If you already have a Huawei Cloud account, skip this step. If you do not have one, do as follows:

   a. Log in to the **Huawei Cloud official website**, and click **Sign Up**.

   b. Sign up for a HUAWEI ID as prompted. For details, see **Signing Up for a HUAWEI ID and Enabling Huawei Cloud Services**.

      Your personal information page is displayed after the registration completes.

   c. Complete **real-name authentication** for your individual or enterprise account.

2. **Grant permissions for users.**

   If you log in to the console using a Huawei Cloud account, skip this step.

   If you log in to the console as an IAM user, first contact your CTS administrator (account owner or a user in the **admin** user group) to obtain the **CTS FullAccess** permissions. For details, see **Assigning Permissions to an IAM User**.

Viewing Traces

After you enable CTS and the management tracker is created, CTS starts recording operations on cloud resources. After a data tracker is created, CTS starts recording user operations on data in OBS buckets. CTS retains operation records of the latest seven days.

This section describes how to query and export operation records of the last seven days on the CTS console.

Viewing Real-Time Traces in the Trace List of the New Edition

Step 1 Log in to the **CTS console**.

Step 2 In the navigation pane, choose **Trace List**.

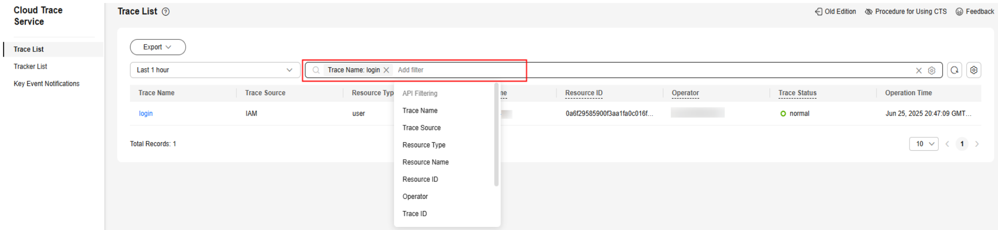
Step 3 In the time range drop-down list above the trace list, select a desired query time range: **Last 1 hour**, **Last 1 day**, or **Last 1 week**. You can also specify a custom time range within the last seven days.

Step 4 The search box above the trace list supports advanced queries. Combine one or more filters to refine your search.

Table 15-2 Trace filtering parameters

Parameter	Description
Read-Only	After selecting the **Read-Only** filter, you can select either **Yes** or **No** from the drop-down list. ● **Yes**: filters read-only operation traces, for example, resource query operations. This option is available after **Read-Only Trace Reporting** has been enabled in the **Configuration Center** and at least one read-only trace has been triggered. ● **No**: filters non-read-only operation traces, such as creating, modifying, and deleting resources.
Trace Name	Name of a trace. The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported. For details about the operations that can be audited for each cloud service, see **Supported Services and Operations**. Example: **updateAlarm**
Trace Source	Cloud service name abbreviation. The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported. Example: **IAM**
Resource Name	Name of a cloud resource involved in a trace. The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported. If the cloud resource involved in the trace does not have a resource name or the corresponding API operation does not involve the resource name parameter, leave this field empty. Example: **ecs-name**

Parameter	Description
Resource ID	ID of a cloud resource involved in a trace. The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported. Leave this field empty if the resource has no resource ID or if resource creation failed. Example: *{VM ID}*
Trace ID	Value of the **trace_id** parameter for a trace reported to CTS. The entered value requires an exact match. Fuzzy matching is not supported. Example: **01d18a1b-56ee-11f0-ac81-******1e229**
Resource Type	Type of a resource involved in a trace. The entered value is case-sensitive and requires an exact match. Fuzzy matching is not supported. For details about the resource types of each cloud service, see **Supported Services and Operations**. Example: **user**
Operator	User who triggers a trace. Select one or more operators from the drop-down list. If the value of **trace_type** in a trace is **SystemAction**, the operation is triggered by the service and the trace's operator may be empty. For details about the relationship between IAM identities and operators and the operator username format, see **Relationship Between IAM Identities and Operators**.
Trace Status	Select one of the following options from the drop-down list: ● **normal**: The operation succeeded. ● **warning**: The operation failed. ● **incident**: The operation caused a fault that is more serious than a normal failure, for example, causing other faults.
Enterprise Project ID	ID of the enterprise project to which a resource belongs. To check enterprise project IDs, go to the Enterprise Project Management Service (EPS) console and choose **Project Management** in the navigation pane. Example: **b305ea24-c930-4922-b4b9-******1eb2**
Access Key	Temporary or permanent access key ID. To check access key IDs, hover over your username in the upper right corner of the console and select **My Credentials** from the pop-up list. On the displayed page, choose **Access Keys** in the navigation pane. Example: **HSTAB47V9V*******TLN9**



Step 5 On the **Trace List** page, you can also export and refresh the trace list, and customize columns to display.

- Enter any keyword in the search box and press **Enter** to filter desired traces.
- Click **Export** to export all traces in the query result as an .xlsx file. The file can contain up to 5,000 records.
- Click to view the latest information about traces.
- Click to customize the information to be displayed in the trace list. If **Auto wrapping** is enabled ( ), excess text will move down to the next line; otherwise, the text will be truncated. By default, this function is disabled.

Step 6 (Optional) On the **Trace List** page of the new edition, click **Old Edition** in the upper right corner to switch to the **Trace List** page of the old edition.

----End

Viewing Traces in the Trace List of the Old Edition

Step 1 Log in to the **CTS console**.

Step 2 In the navigation pane, choose **Trace List**.

Step 3 Each time you log in to the CTS console, the new edition is displayed by default. Click **Old Edition** in the upper right corner to switch to the trace list of the old edition.

Step 4 In the upper right corner of the page, select a desired query time range: **Last 1 hour**, **Last 1 day**, or **Last 1 week**. You can also specify a custom time range within the last seven days.

Step 5 Set filters to search for your desired traces.

Table 15-3 Trace filtering parameters

Parameter	Description
Trace Type	Select **Management** or **Data**. ● Management traces record operations performed by users on cloud service resources, including creation, modification, and deletion. ● Data traces are reported by OBS and record operations performed on data in OBS buckets, including uploads and downloads.

Parameter	Description
Trace Source	Select the name of the cloud service that triggers a trace from the drop-down list.
Resource type	Select the type of the resource involved in a trace from the drop-down list. For details about the resource types of each cloud service, see **Supported Services and Operations**.
Operator	User who triggers a trace. Select one or more operators from the drop-down list. If the value of **trace_type** in a trace is **SystemAction**, the operation is triggered by the service and the trace's operator may be empty. For details about the relationship between IAM identities and operators and the operator username format, see **Relationship Between IAM Identities and Operators**.
Trace Status	Select one of the following options: ● **Normal**: The operation succeeded. ● **Warning**: The operation failed. ● **Incident**: The operation caused a fault that is more serious than a normal failure, for example, causing other faults.

Step 6 Click **Query**.

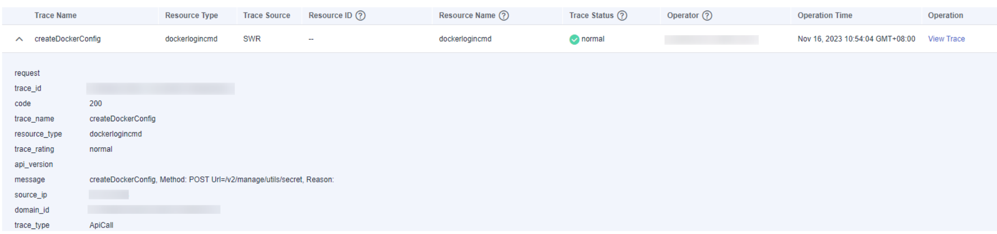
Step 7 On the **Trace List** page, you can also export and refresh the trace list.

- Click **Export** to export all traces in the query result as a CSV file. The file can contain up to 5,000 records.
- Click to view the latest information about traces.

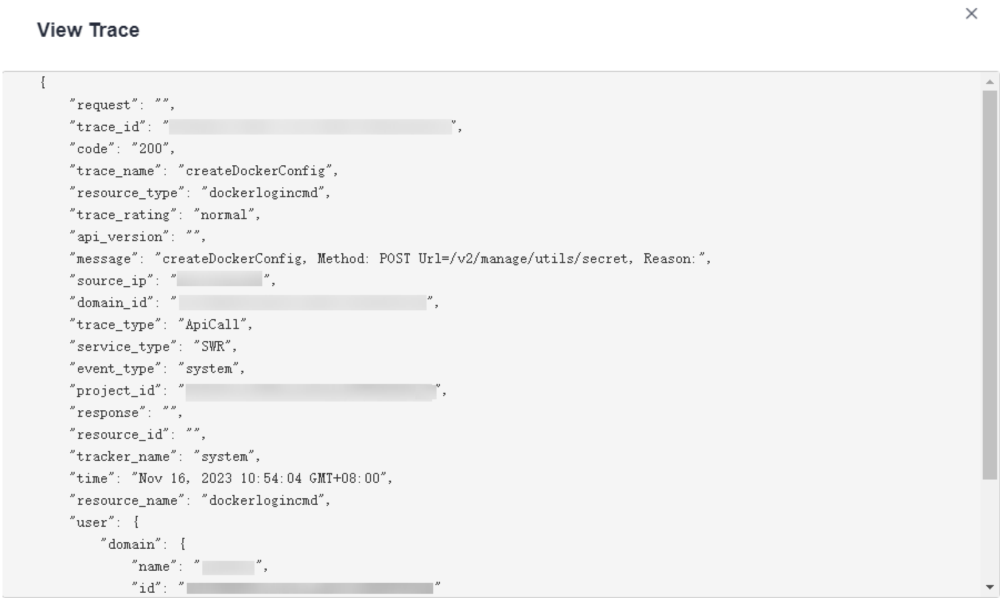
Step 8 In the **Tampered or Not** column of a trace, check whether the trace is tampered with.

- **No**: The trace is not tampered with.
- **Yes**: The trace is tampered with.

Step 9 Click on the left of a trace to expand its details.



Step 10 Click **View Trace** in the **Operation** column. The trace details are displayed.



Step 11 (Optional) On the **Trace List** page of the old edition, click **New Edition** in the upper right corner to switch to the **Trace List** page of the new edition.

----End

Helpful Links

- For details about the key fields in the trace structure, see **Trace Structure** and **Example Traces**.
- You can use the following examples to learn how to query a specific trace:
  - Use CTS to audit Elastic Volume Service (EVS) creation and deletion operations from the last two weeks. For details, see **Security Auditing**.
  - Use CTS to locate a fault or creation failure for an Elastic Cloud Server (ECS). For details, see **Fault Locating**.
  - Use CTS to check all operation records for an ECS. For details, see **Resource Tracking**.

16 Reference

16.1 Header Description

Table 16-1 Header Description

Protocol Header Name	Description
Accept	Acceptable types of response contents.
Accept-Charset	Acceptable character sets.
Accept-Datetime	Acceptable versions displayed by time.
Accept-Encoding	Acceptable encoding methods.
Accept-Language	Acceptable natural languages of response contents.
Authorization	Information used for HTTP authentication.
Cache-Control	Instructions that must be followed by all cache mechanisms for a request or response chain.
Connection	Priority connection types for a browser.
Content-Length	The length of the request body represented by 8-byte arrays.
Content-MD5	The 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).
Cookie	An HTTP cookie sent by servers through **Set-Cookie**.
Date	The date and time when a message is sent.

Protocol Header Name	Description
DNT	A request for a web application to stop tracking a user. In the Firefox browser, it is equivalent to the X-Do-Not-Track protocol header field (supported for Firefox/4.0 Beta 11 or later). Safari and Internet Explorer 9 also support this field.
Expect	Specific actions required by a client for a server.
Front-End-Https	Refers to non-standard header fields used by Microsoft servers and load balancers.
From	The email address of the user who initiates the request.
Host	Domain name of the server (used for virtual host), and port number of the transmission control protocol listened to by the server. If the requested port is the standard port of the corresponding service, the port number can be omitted. This field is mandatory since HTTP/1.1. If the domain name in the URL is an IP address, this field is automatically added. Otherwise, enter the IP address and port number of the tested application in this field.
If-Match	The corresponding operation is performed only when the entity provided by the client matches the entity on the server. It is used in a method such as **PUT** to update a resource which has not been modified since the last update.
If-Modified-Since	Returning of **304 Not Modified** is allowed when the corresponding content is not modified.
If-None-Match	Returning of **304 Not Modified** is allowed when the corresponding content is not modified. Refer to the HTTP entity tag. In a typical use, when a URL is requested, the web server returns the resource and its corresponding **ETag** value, which is placed in the **ETag** field of the HTTP. The client can then decide whether to cache the resource and its **ETag**. If the client wants to request the same URL again, it sends a request that contains the saved **ETag** and the **If-None-Match** field.
If-Range	If an entity is not modified, send one or more parts that are missing to the sender. Otherwise, send the entire new entity.
If-Unmodified-Since	A response is sent only when the entity has not been modified since a specific time.
Max-Forwards	Refers to the number of times a message can be forwarded by the proxy and gateway.
Origin	Refers to a sharing request initiated for resources of different origins. The server is required to add an **Access-Control-Allow-Origin** field to the response.

Protocol Header Name	Description
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.
Proxy-Connection	Derives from the errors in the implementation of an early HTTP version. The function of this field is the same as that of the standard Connection field.
Proxy-Password	Proxy password.
Proxy-Server	Proxy service.
Proxy-Username	Proxy username.
Range	Requests only a part of an entity with the byte offset starting from 0.
Referer	Refers to the previous page accessed by a browser. A link on this page brings the browser to the currently requested page.
TE	Refers to the transmission coding mode expected by a browser. You can use a value of **Transfer-Encoding** in the response protocol header. In addition, the value **trailers** (related to the block transmission mode) indicates that the browser expects to receive additional fields if the size of the last block is 0.
Upgrade	The server needs to be upgraded to another protocol.
User-Agent	Refers to the character string of the browser identity.
Via	Refers to request-sending agents informed to a server.
Warning	Refers to a general warning indicating that errors may exist in the body of an entity.
X-Wap-Profile	Refers to an XML file linked to the Internet. The file describes devices being connected.
X-Requested-With	Used to identify Ajax and XML requests. Most JavaScript frameworks send this field and set its value to **XMLHttpRequest**.
X-Http-Method-Override	A web application is requested to use the method (usually PUT or DELETE) specified in the protocol header field to override the method specified in the request (usually POST). Use this method when a browser or firewall prevents direct sending of the PUT or DELETE method (may be caused by a vulnerability in the software, which needs to be fixed, or because a configuration option is required, in which case the situation should not be bypassed).

Protocol Header Name	Description
X-Forwarded-Proto	Refers to a fact standard used to identify the protocol used at the beginning of an HTTP request.
X-Forwarded-Host	Refers to a fact standard used to identify the host request header originally sent by the client.
X-Forwarded-For	Refers to a fact standard used to identify the original Internet address of a client that is connected to a web server through an HTTP agent or load balancer.
X-Csrf-Token	The **X-CSRFToken** or **X-XSRF-TOKEN** header is used to prevent cross-site request forgery.
X-ATT-DeviceId	Enables the server to easily interpret the common device models and firmware information in the **User-Agent** field of the AT&T device.

16.2 Regular Expression Metacharacters

Table 16-2 Metacharacter description

Metacharacter	Description
.	Matches any characters except **\n**. If **\n** needs to be included, use other modes such as **[\s\S]**.
^	Matches the start position of an input character string and does not match any characters. Use **\^** to match the character itself.
$	Matches the end position of an input character string and does not match any characters. Use **\$** to match the character itself.
*	The preceding characters or sub-expressions are matched zero or more times. ***** is equivalent to {0,}. For example, **\^*b** can match **b**, **^b**, **^^b**, and so on.
+	Matches preceding characters or sub-expressions one or more times, equivalent to **{1,}**. For example, **a+b** can match **ab**, **aab**, **abb**, **aaab**, and so on.
?	Matches preceding characters or sub-expression zero times or once, equivalent to **{0,1}**. For example, **a[cd]?** can match **a**, **ac**, and **ad**. When this character follows any other qualifier such as *****, **+**, **?**, **{n}**, **{n,}**, or **{n,m}**, the matching mode is **non-greedy**. **Non-greedy** mode matches the shortest possible searched character strings, and the default **greedy** mode matches the longest possible searched character strings. For example, the character string **oooo**, **o+?** matches only a single **o**, while **o+** matches all **o**.

<table>
<tr><th>Metacharacter</th><th>Description</th></tr>
<tr><td>|</td><td>The logic or is performed on two matching conditions. For example, the regular expression (him|her) matches it belongs to him and it belongs to her, but cannot match it belongs to them.</td></tr>
<tr><td>\</td><td>Marks the next character as a special character, text, reverse reference, or octal escape character. For example, n matches the character n, \n matches the newline character, \\ matches \, and \(matches (.</td></tr>
<tr><td>\w</td><td>Matches a letter, digit, or underscore (_).</td></tr>
<tr><td>\W</td><td>Matches any character that is not a letter, digit, or underscore (_).</td></tr>
<tr><td>\s</td><td>Matches any blank character, such as a space, tab character, or form feed. It is equivalent to [\f\n\r\t\v].</td></tr>
<tr><td>\S</td><td>Matches any character except blank characters and is equivalent to [^\f\n\r\t\v].</td></tr>
<tr><td>\d</td><td>Matches any digit and is equivalent to [0–9].</td></tr>
<tr><td>\D</td><td>Matches any non-digital character and is equivalent to [^0-9].</td></tr>
<tr><td>\b</td><td>Matches a word boundary (the position between a word and a space) and does not match any characters. For example, er\b matches er in never but does not match er in verb.</td></tr>
<tr><td>\B</td><td>A non-word boundary match. For example, er\B matches er in verb but does not match er in never.</td></tr>
<tr><td>\f</td><td>Matches a form feed and is equivalent to \x0c and \cL.</td></tr>
<tr><td>\n</td><td>Matches a linefeed and is equivalent to \x0a and \cJ.</td></tr>
<tr><td>\r</td><td>Matches a carriage return character and is equivalent to \x0d and \cM.</td></tr>
<tr><td>\t</td><td>Matches a tab character and is equivalent to \x09 and \cI.</td></tr>
<tr><td>\v</td><td>Matches a vertical tab character and is equivalent to \x0b and \cK.</td></tr>
<tr><td>\cx</td><td>Matches control characters indicated by x. For example, if \cM matches Control-M or a carriage return character, the value of x must be between A–Z or a–z. Otherwise, the c character indicates c itself.</td></tr>
<tr><td>{n}</td><td>The value n is a non-negative integer and refers to the number of matching times. For example, o{2} does not match o in Bob, but matches o in food.</td></tr>
</table>

<table>
<tr><th>Metacharacter</th><th>Description</th></tr>
<tr><td>{n,}</td><td>The value n is a non-negative integer and refers to the minimum number of matching times. For example, o{2,} does not match o in Bob but matches all o in foooood. o{1,} is equivalent to o+, and o{0,} is equivalent to o*.</td></tr>
<tr><td>{n,m}</td><td>The values n and m (n≤m) are non-negative integers, where n refers to the minimum number of matching times and m refers to the maximum matching times. For example, o{1,3} matches the first three os in fooooood, and o{0,1} is equivalent to o?. Note that a space cannot be inserted between commas and digits. For example, ba{1,3} can match ba, baa, or baaa.</td></tr>
<tr><td>x|y</td><td>Matches x or y. For example, z|food matches z or food, and (z|f)ood matches zood or food.</td></tr>
<tr><td>[xyz]</td><td>Refers to a character set that matches any characters included. For example, [abc] matches a in plain.</td></tr>
<tr><td>[^xyz]</td><td>Refers to a reverse character set that matches any characters except xyz. For example, [^abc] matches p in plain.</td></tr>
<tr><td>[a-z]</td><td>Refers to a character range and matches any characters in the specified range. For example, [a-z] matches any lowercase letters from a to z.</td></tr>
<tr><td>[^a-z]</td><td>Refers to a reverse character range and matches any characters not in the specified range. For example, [^a-z] does not match any characters from a to z.</td></tr>
<tr><td>( )</td><td>Defines expressions between (and) as group and save characters that match the expression to a temporary area. A maximum of nine characters can be saved in a regular expression, and these characters can be referenced by symbols \1 to \9.</td></tr>
<tr><td>(pattern)</td><td>Matches pattern and captures sub-expressions of the match. You can use the $0–$9 attribute to retrieve captured matches from the result match set.</td></tr>
<tr><td>(?:pattern)</td><td>Matches pattern but does not capture sub-expressions of the match. That is, it is a non-capture match and does not store matches for future use. This is useful for the or character combined with (|). For example, industr(?:y|ies) is a simpler expression than industry|industries.</td></tr>
</table>

<table>
<tr><th>Metacharacter</th><th>Description</th></tr>
<tr><td>(?=pattern)</td><td>Refers to a non-capture match and indicates a forward positive pre-check, searching character strings at the start position of any character strings that match pattern. There is no need to capture the match for future use. For example, "Windows(?=95|98|NT|2000)" matches "Windows" in "Windows2000", but does not match "Windows" in "Windows3.1". A pre-check does not consume characters. That is, after a match occurs, the next search starts immediately, instead of starting from pre-checked characters.</td></tr>
<tr><td>(?!pattern)</td><td>Refers to a non-capture match and indicates a forward negative pre-check, searching character strings at the start position of any character strings that do not match pattern. There is no need to capture the match for future use. For example, "Windows(?=95|98|NT|2000)" matches "Windows" in "Windows3.1", but does not match "Windows" in "Windows2000".</td></tr>
</table>

To match special characters, add **** before the special characters. For example, to match the following special characters: **^**, **$**, **()**, **[]**, **{}**, **.**, **?**, **+**, *****, and **|**, use **\^**, **\$**, **\ (**, **\)**, **\ [**, **\]**, **\{**, **\}**, **\.**, **\?**, **\+**, *****, and **\|**.

16.3 HTTP Status Codes

Table 16-3 HTTP status codes

Status Code	Message	Description
100	Continue	The initial part of a request has been received and has not yet been rejected by the server.
101	Switching Protocols	The server understands and is willing to comply with the client's request, via the **Upgrade** header field, for a change in the application protocol being used on this connection.
102	Processing	This status code is an interim response used to inform the client that the server has accepted the complete request, but has not yet completed it. This status code SHOULD only be sent when the server has a reasonable expectation that the request will take significant time to complete.
103	Early Hints	This status code indicates to the client that the server is likely to send a final response with the header fields included in the informational response.

Status Code	Message	Description
200	OK	The request has succeeded.
201	Created	The request has been fulfilled and has resulted in one or more new resources being created.
202	Accepted	The request has been accepted for processing, but the processing has not been completed.
203	Non-Authoritative Information	The request was successful but the enclosed content has been modified from that of the origin server's **200 (OK)** response by a transforming proxy.
204	No Content	The server has successfully fulfilled the request and that there is no additional content to send in the response content.
205	Reset Content	The server has fulfilled the request and desires that the user agent reset the "document view", which caused the request to be sent, to its original state as received from the origin server.
206	Partial Content	The server is successfully fulfilling a range request for the target resource by transferring one or more parts of the selected representation.
207	Multi-Status	This status code provides status for multiple independent operations.
208	Already Reported	This status code can be used inside a **DAV: propstat** response element to avoid enumerating the internal members of multiple bindings to the same collection repeatedly.
226	IM Used	The server has fulfilled a GET request for the resource, and the response is a representation of the result of one or more instance-manipulations applied to the current instance.
300	Multiple Choices	The target resource has more than one representation, each with its own more specific identifier, and information about the alternatives is being provided so that the user (or user agent) can select a preferred representation by redirecting its request to one or more of those identifiers.
301	Moved Permanently	The target resource has been assigned a new permanent URI and any future references to this resource ought to use one of the enclosed URIs.
302	Found	The target resource resides temporarily under a different URI.

Status Code	Message	Description
303	See Other	The server is redirecting the user agent to a different resource, as indicated by a URI in the **Location** header field, which is intended to provide an indirect response to the original request.
304	Not Modified	A conditional GET or HEAD request has been received and would have resulted in a **200 (OK)** response if it were not for the fact that the condition evaluated to false.
305	Use Proxy	The **305 (Use Proxy)** status code was defined in a previous version of this specification and is now deprecated.
307	Temporary Redirect	The target resource resides temporarily under a different URI and the user agent MUST NOT change the request method if it performs an automatic redirection to that URI.
308	Permanent Redirect	The target resource has been assigned a new permanent URI and any future references to this resource ought to use one of the enclosed URIs.
400	Bad Request	The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).
401	Unauthorized	The request has not been applied because it lacks valid authentication credentials for the target resource.
402	Payment Required	The status code is reserved for future use.
403	Forbidden	The server understood the request but refuses to fulfill it. A server that wishes to make public why the request has been forbidden can describe that reason in the response content (if any).
404	Not Found	The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.
405	Method Not Allowed	The method received in the request-line is known by the origin server but not supported by the target resource.
406	Not Acceptable	The target resource does not have a current representation that would be acceptable to the user agent, according to the proactive negotiation header fields received in the request, and the server is unwilling to supply a default representation.

Status Code	Message	Description
407	Proxy Authentication Required	This status code is similar to **401 (Unauthorized)**, but it indicates that the client needs to authenticate itself in order to use a proxy for this request.
408	Request Timeout	The server did not receive a complete request message within the time that it was prepared to wait.
409	Conflict	The request could not be completed due to a conflict with the current state of the target resource. This code is used in situations where the user might be able to resolve the conflict and resubmit the request.
410	Gone	The target resource is no longer available at the origin server and that this condition is likely to be permanent. If the origin server does not know, or has no facility to determine, whether or not the condition is permanent, the status code **404 (Not Found)** ought to be used instead.
411	Length Required	The server refuses to accept the request without a defined **Content-Length**.
412	Precondition Failed	One or more conditions given in the request header fields evaluated to false when tested on the server.
413	Content Too Large	The server is refusing to process a request because the request content is larger than the server is willing or able to process.
414	URI Too Long	The server is refusing to service the request because the target URI is longer than the server is willing to interpret.
415	Unsupported Media Type	The origin server is refusing to service the request because the content is in a format not supported by this method on the target resource.
416	Range Not Satisfiable	The set of ranges in the request's **Range** header field has been rejected either because none of the requested ranges are satisfiable or because the client has requested an excessive number of small or overlapping ranges (a potential denial of service attack).
417	Expectation Failed	The expectation given in the request's **Expect** header field could not be met by at least one of the inbound servers.
421	Misdirected Request	The request was directed at a server that is unable or unwilling to produce an authoritative response for the target URI.

Status Code	Message	Description
422	Unprocessable Content	The server understands the content type of the request content (hence a **415 (Unsupported Media Type)** status code is inappropriate), and the syntax of the request content is correct, but it was unable to process the contained instructions.
423	Locked	The source or destination resource of a method is locked.
424	Failed Dependency	The method could not be performed on the resource because the requested action depended on another action and that action failed.
425	Too Early	The server is unwilling to risk processing a request that might be replayed.
426	Upgrade Required	The server refuses to perform the request using the current protocol but might be willing to do so after the client upgrades to a different protocol.
428	Precondition Required	The origin server requires the request to be conditional.
429	Too Many Requests	The user has sent too many requests in a given amount of time ("rate limiting").
431	Request Header Fields Too Large	The server is unwilling to process the request because its header fields are too large.
451	Unavailable For Legal Reasons	The server is denying access to the resource as a consequence of a legal demand.
500	Internal Server Error	The server encountered an unexpected condition that prevented it from fulfilling the request.
501	Not Implemented	The server does not support the functionality required to fulfill the request.
502	Bad Gateway	The server, while acting as a gateway or proxy, received an invalid response from an inbound server it accessed while attempting to fulfill the request.
503	Service Unavailable	The server is currently unable to handle the request due to a temporary overload or scheduled maintenance, which will likely be alleviated after some delay.
504	Gateway Timeout	The server, while acting as a gateway or proxy, did not receive a timely response from an upstream server it needed to access in order to complete the request.

Status Code	Message	Description
505	HTTP Version Not Supported	The server does not support, or refuses to support, the major version of HTTP that was used in the request message.
506	Variant Also Negotiates	The server has an internal configuration error: the chosen variant resource is configured to engage in transparent content negotiation itself, and is therefore not a proper end point in the negotiation process.
507	Insufficient Storage	The method could not be performed on the resource because the server is unable to store the representation needed to successfully complete the request.
508	Loop Detected	The server terminated an operation because it encountered an infinite loop while processing a request with **Depth: infinity**.
511	Network Authentication Required	The client needs to authenticate to gain network access.

16.4 Parameters in Exported PerfTest Project Files

Table 16-4 Request composition of think time

Parameter	Description
name	Can be modified.
t	*Duration (ms) **Think Time** refers to the waiting time between two consecutive operations performed by a user, such as the interval between login and search. Suppose that the response time for each running of a transaction is 0.5s. ● To execute two transaction requests per second, do not set the **Think Time** request composition. ● To execute only one transaction request per second, set **Think Time** to **1s**. If the think time is set to 1s and the response time is longer than 1s, the think time will not take effect, and pressure test requests are sent according to the response time.

Table 16-5 Request composition of packet

Parameter	Description
name	Can be modified.
http_version	Protocol type. HTTP, HTTPS, TCP, and UDP protocols are supported.
When **Protocol Type** is **HTTP** or **HTTPS**, set the following parameters.	
method	GET, POST, PATCH, PUT, and DELETE are supported.
return_timeout	Timeout period for waiting for a response from the server when a request is sent. If this parameter is not set, the default response timeout interval (5,000 ms) is used.
URL	URL for sending a request, for example, **http://***domain name***/***path*, or **http://***domain name***/path?key1=value1&key2=value2**.
headers	Headers consist of keywords and values and are used to notify servers of clients' requests. For details about headers, see **Header Description**.
Packet content	The body of an entity contains a data block consisting of random data. Not all packets contain the body of an entity. If global variables have been set or local variables have been set for response extraction, variables can be used in the packet content. During a pressure test, variables in the packet content will be replaced with specified values.
When **Protocol Type** is **TCP**, set the following parameters. TCP packets do not support the response extraction and checkpoint functions.	
IP	IP address of the tested server to which requests are sent.
port	Port number of the tested server to which requests are sent.
connect_timeout	Timeout duration for the server's response after a connection is initiated.
return_timeout	Timeout duration for waiting for the server's response after a connection is established.
Connection Settings	• **Repeated use**: When a request response is complete, the connection remains and is used to send and receive the next request response. • **Close**: When a request response is complete, the connection is closed and re-established next time.

Parameter	Description
check_end_type	Used to judge whether a request response has been received. It is recommended that a unique end-of-text character be set. If multiple such characters are present in a response, the response is considered complete when the first character is received. As a result, the received response data may be incomplete. ● **Length of returned data**: length of the returned data, in bytes. When a response of this length is received, data receiving is complete. ● **End-of-text character**: ending mark of the returned data. When an end-of-text character is received, data receiving is complete.
body	The body of an entity contains a data block consisting of random data. Not all packets contain the body of an entity. Content format: Select character strings or a hexadecimal code stream based on the service request content of the tested server. A hexadecimal code stream can contain only numbers 0–9 and letters a–f. The total number of characters must be an even number. If global variables have been set or local variables have been set for response extraction, variables can be used in the packet content. During a pressure test, variables in the packet content will be replaced with specified values.

Table 16-6 Request composition of response extraction

Parameter	Description
name	Must be unique. The value of the response extraction is assigned to this variable.
range	Range of the content to be extracted. ● Packet content ● Header ● URL ● Response Code The packet content, header, and URL can be extracted using regular expressions.

Parameter	Description
regexp	A regular expression specifies a type of logical expressions performed on strings. In a regular expression, you use certain predefined strings and a combination of these strings to form a rule string that is used to specify a specific filtering logic. A complete regular expression consists of two types of characters: special characters (also called meta characters) and literal or common text characters (such as letters, digits, and underscores). For details about meta characters, see **Regular Expression Metacharacters**. Parentheses **()** indicate extraction and are used to enclose content to be extracted. The content in a pair of parentheses forms a sub-expression.
match_index	This parameter indicates the sequence number of the matched content extracted through a regular expression. This parameter cannot be set to **0**. Value: a positive integer
exp_index	This parameter indicates the sequence number of the parsed sub-expression. ● Value **0** indicates that the entire regular expression is matched. ● Value **1** indicates that the first sub-expression of the regular expression is matched, that is, the content extracted by the first **()**. Value range: natural numbers After extracting content through **Regular Expression** and **Sequence Number of Matching Item**, use **Expression Value** to obtain the final content.
JSON Key Name	Enter the key name to be obtained. For example, {key:{"key1":"v1","key2":["v2","v3"]}}. If you want to obtain **v2**, enter **key.key2[0]**.
default	Indicates the value returned when a regular expression match fails.

Table 16-7 Request composition of checkpoint

Parameter	Description
name	Can be modified.
value	This parameter is used to respond to HTTP, HTTPS, TCP and UDP response status codes carried in a packet, including 1XX, 2XX, 3XX, 4XX, and 5XX.

Parameter	Description
header_checks	Headers of the HTTP, HTTPS, TCP, and UDP.
body_checks	Body parts of HTTP, HTTPS, TCP, MQTT, and UDP, namely the load parts of the requests and responses.

Table 16-8 Adding a test project

Parameter	Description
name	Indicates the test project name.
description	Indicates description of the test project.

Table 16-9 Adding a transaction

Parameter	Description
name	Indicates the transaction name.
contents	Request content. You can add transaction requests under a transaction based on service requirements.

Table 16-10 Adding a task

Parameter	Description
issue_num	Number of concurrent users. The number of concurrent users refers to the number of users performing operations on the system at the same time. In CodeArts PerfTest, it is the number of virtual users set when you define test task phases.
name	Phase name. Set a name for the service scenario, for example, **homepage test**.
time	Duration (second). The maximum time for performing a pressure test in the current phase.
count	Total number of transmissions. During the running of a task, the transaction is calculated based on the number of running times. When the specified value is met, the performance test of the transaction under the task is terminated.

Table 16-11 Adding a global variable

Parameter	Description
name	Global variable name.
variable	Global variable value. Variable content is transmitted in plain text during the pressure test. Exercise caution when entering the content.
variable_type	Global variable type. When the variable type is **Integer**, enter the value range of the variable. A pressure test task reads the value range of the corresponding global variable when it is running. For each virtual user, the variable value is polled. That is, the first virtual user obtains the first value of the variable, and the second virtual user obtains the second value. If the values are exhausted, the next virtual user obtains the first value again. Add multiple variables as required. A pressure test task reads the value range of the corresponding global variable when it is running. For each virtual user, the variable value is polled. That is, the first virtual user obtains the first value of the variable, and the second virtual user obtains the second value. If the values are exhausted, the next virtual user obtains the first value again.

16.5 Mapping Between JMeter and PerfTest Fields

CodeArts PerfTest supports the import and automatic switchover of the JMeter script. **Table 16-12** describes the mapping between JMeter and PerfTest fields.

Table 16-12 Mapping between JMeter and PerfTest fields

<table>
<tr><th colspan="2">JMeter Field</th><th>PerfTest Field</th><th>Parameter Description</th></tr>
<tr><td rowspan="3">Thread group</td><td>Name</td><td>Transaction Name/ Task Name</td><td rowspan="3">The HTTP request and the subsequent HTTP request form a transaction.
If no valid value is obtained, number of concurrent requests is 1 and the duration is 1 minute by default.</td></tr>
<tr><td>Number of threads</td><td>Number of concurrent test tasks</td></tr>
<tr><td>Duration</td><td>Test task duration</td></tr>
</table>

JMeter Field		PerfTest Field	Parameter Description
User-defined variables	Name	Enumeration variable name	Global variable, enumerated type. The name must be unique.
	Value	Enumeration variable value	
User parameters	Name	Enumeration variable name	Global variable, enumerated type. The name must be unique.
	User_1	Enumeration variable value	
	User_2	Enumeration variable value	
	...	Enumeration variable value	
	User_N	Enumeration variable value	
HTTP header manager	Name	Packet header	When the POST, PATCH, PUT, or DELETE request mode does not contain the **Content-Type** header, the **Content-Type** header value is set to **application/x-www-form-urlencoded** by default.
	Value	Header value	
Default HTTP request	Protocol	-	If the HTTP request does not contain a value, the default value of the HTTP request is used. Priority: HTTP request > thread group > test plan.
	Server name or IP address		
	Port number		
	Path		
	Parameter		
	Message body data		
	Response timeout in the advanced options.		

JMeter Field		PerfTest Field	Parameter Description
HTTP request	Protocol	Protocol type	Packet. Ignore request modes except POST, GET, PATCH, PUT, and DELETE. The default value is 5,000 ms when no response timeout value is obtained.
	Protocol, server name or IP address, port number, and path	Request URL	
	Method	Request mode	
	Parameter	Add the content to the URL or packet content as required.	
	Message body data	Packet content	
	Response timeout in the advanced options.	Response timeout	
Regular expression extractor (the response field to be checked)	Body	Packet content	Response extraction. Only the regular expression extractor and fixed timer under the HTTP requests can be imported. If the default value is empty, the reference name is used by default.
	Body (unescaped)	Packet content	
	Body as a Document	Packet content	
	Message header	Header	
	URL	URL	
	Response code	Response code	
	Request Headers. Response information.	Import is not supported. The regular expression extractor is ignored.	
Regular expression extractor	Reference name	Variable name	
	Regular expression	Regular expression	
	Template	Sequence number of matching item	
	Match digits	Expression value	
	Default value	Default value	
Fixed timer	Thread latency	Duration	Think time. If no valid value is read, the default value 1,000 is used.

JMeter Field		PerfTest Field	Parameter Description
Random function	-	Random number in a range	An integer generated randomly within the range entered by a user. For example, ${__Random(-2147483648,2147483647)}
JSON extractor	Name of created variables	Variable name	Response extraction. Default extraction range: parameter values in .json format. Only the JSON extractor under the HTTP requests can be imported.
	JSON Path expressions	Key name	
	Default Values	Default value	

