

# Tencent Smart Advisor–Chaotic Fault Generator Quick Start Product Documentation



## Copyright Notice

©2013–2026 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice

 Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by the Tencent corporate group, including its parent, subsidiaries and affiliated companies, as the case may be. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

# Contents

## Quick Start

Quick Start with the Console

Quick Start with API

# Quick Start

## Quick Start with the Console

Last updated: 2024-09-26 15:34:19

This document describes how to quickly get started with Tencent Smart Advisor-Chaotic Fault Generator.

### Introduction

1. **Overview:** Provide features such as beginner operation guides, access to the event experience, and popular experiment templates to help you obtain product information in a timely manner.
2. **Experiment Management:** Create a new experiment and manage all historical experiments.
3. **Action Library Management:** View details of the platform's fault action library and manage custom action scripts.
4. **Template Library Management:** View platform-recommended experiment templates and manage custom template library.
5. **Agent Management:** Manage the agents pre-installed in the cluster used for TKE-type fault actions.

### Directions

To validate the system's fault tolerance, availability, and other performance metrics, you can inject appropriate faults into the system and observe its behavior. This allows you to identify potential issues within the system and address them promptly. Let's take the High CPU Utilization experiment as an example to demonstrate how to quickly create a chaos engineering experiment.

### Step 1: Create an Experiment

1. Log in to the [Tencent Smart Advisor > Chaotic Fault Generator](#), go to the **Experiment Management** page, and click **Create a New Experiment**.
2. When creating a new experiment, you can either select a platform-recommended **Industry Experience Library** or select **Skip and create a blank experiment**. If you create the experiment using a template, the basic experiment information and failure action orchestration details will be automatically populated. You only need to select the instance resources. You can choose to skip this step.
3. Go to the **Basic Information Filling** page, and fill in **Experiments Name**, **Experiments Description** and **Tag**. Among them, **Tag** can be used to manage and search for experiments. Click **Next**.
4. Go to the **Experiment Object Configuration** page, fill in the **Action Group** orchestration details, and select **Add Instance**.
  - Within the same action group, fault injection can only be performed on the same object type.

- You can add multiple actions within an action group as needed, allowing for flexible combination and orchestration.
5. Click **Add Instance**, and select the instance resources where you want to inject faults, such as selecting a CVM instance.
    - You can search by instance type and instance name.
    - Batch addition of instances is supported.
  6. After completing **adding instances**, click **Add an experiment action**.
  7. Click **Add Now** to add an experiment action, such as selecting the High CPU Utilization fault action under the CPU Resources category, and then click **Next**.
    - The platform supports searching for a variety of fault atomic actions.
    - Custom fault injection can be achieved by uploading custom scripts, allowing you to meet specific business needs.
  8. **Set action parameters**, including configuring **wait time before and after the action** and **timeout period** under **General Parameters** to control the experiment's pace.

Set the **Duration** for this action in the **Action Parameters**. After the setting is completed, click **Confirm** to complete the action addition.
  9. Click **Next** to proceed to the **Global Configuration** page, where you can select the action **Execution Method**, and configure **Guardrail Policy** and **Monitoring Metrics**.
    - Click **Choose guardrail policy** (this tutorial does not include configuration):
    - Click **Add monitoring metrics**, and select a metric such as **CPU Usage** to observe the fault injection in real-time.
  10. After the configuration is completed, select **Submit** to finalize the experiment. The experiment will be successfully created. The system automatically redirects to the **Experiment Details** page, where a pre-check of the environment is performed. This includes verifying the agent installation, TAT installation status, operating system version, and other relevant factors. The purpose of this check is to ensure that the experiment can proceed smoothly.

**Note:**

The experiment environment check feature is only intended as a risk warning and will not block the experiment process. Even if the environment pre-check does not pass, you can still proceed with the experiment. However, this may lead to experiment failure. To ensure the experiment is executed correctly, it is recommended to follow the pre-check guide before continuing the process.

11. The experiment **Environment Check** has passed. Click **View Detection Details** to view the experiment information.

## Step 2: Execute the Experiment

1. Go to the **Experiment Details** page, and click **Execute** in the upper right corner to start the experiment.

**Note:**

- If during the experiment creation process, the **Automatic Execution** method was selected, the system will automatically start executing the actions after clicking the execution button in the upper right corner, without requiring manual intervention.
- If the execution method is set to **Manual**, after clicking **Execution** in the upper right corner, you will still need to manually click **Start** within the action group to proceed.
- If any action execution fails, the system will automatically switch to **Manual** mode. Manual intervention is required to click **Execute** or **Skip the action** within the action group.

2. In the experiment action group area, select the fault action you want to execute, then click **Execute** to start injecting the fault.
3. During the experiment execution, click an action card to expand and view the action execution details and **View Log**.

### Step 3: End the Experiment

1. Once the fault action is successfully executed, click **End Experiment**.
2. **Fill in Experiment Results**, to document any issues encountered during the experiment, emergency response measures, and other relevant details to facilitate subsequent review and analysis.
3. Click **Generate Experiment Report**, to export a report of the experiment with a single click.  
You can review the CFG experiment report, which includes the following: basic experiment information, experiment action groups, experiment logs, issue records, etc.

### Step 4: Template Library Management

For experiments that need to be conducted frequently or have proven effective in past runs, you can extract the experiment orchestration elements and create them as the template library. The templates can then be quickly reused in future experiments to improve efficiency. Additionally, users can activate or disable custom templates in the **Template Library Management**.

# Quick Start with API

Last updated: 2024-09-26 15:35:06

## Note:

This section primarily demonstrates how to use the official API of CFG to complete an experiment. The API of CFG follows the general API specifications of Tencent Cloud. For details on common parameters and request methods, see [API Invocation Methods](#).

## Creation an Experiment: Create via Template

### Prepare Template ID

#### Method 1: Retrieve from the Console

Log in to the [Chaotic Fault Generator](#), click **Template Library Management**, select the desired template for creating an experiment, and copy the template ID.

#### Method 2: Retrieve via API

Refer to API description in [Query Knowledge Base List](#) to obtain the required **TemplateId** for creating an experiment.

### API Request

Refer to API description in [Create Drill from Experience](#), and make the following request:

```
POST / HTTP/1.1
Host: cfg.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: CreateTaskFromTemplate
<Common request parameters>

{
  "TemplateId": 626, # The template ID
inquired from the previous step
  "TaskConfig": {
    "TaskTitle": "This is an example of creating an experiment from
the API", # Experiment name, if it is not provided, the template na
me will be used by default.
```

```
"TaskGroupsConfig": [  
  {  
    "TaskGroupInstances": [  
      "ins-xxxxxxx" # Instance object  
      ID associated with the action group, such as resource IDs for CVM, CLB,  
      etc.  
    ]  
  }  
]
```

## API Output

```
{  
  "Response": {  
    "RequestId": "f0aee8ac-2ed3-4a7f-a25b-f0d7d228dd30",  
    "TaskId": 3256 # experiment ID  
  }  
}
```

## Create an Experiment: Create via Action

### Prepare Resource Object ID

Refer to API description in [Query Object Type List](#) to obtain the ObjectTypeId for the resource object required in the experiment.

### Prepare Action ID

Refer to API description in [Obtain Action Library List](#) to obtain the ActionId for the actions required in the experiment.

### Prepare Action Parameters

Refer to API description in [Get Action Configuration Parameters](#) to obtain the parameters for the actions required in the experiment.

#### Note:

Here are two parameters provided for creating an action experiment:

- **TaskActionGeneralConfiguration:** General parameter, which is optional. If it is empty, the default action parameter will be used.
- **TaskActionCustomConfiguration:** Custom parameter. Optional parameters have default values set. For required parameters, if the default value is empty, you must explicitly provide a value.

Parameters should be represented in the format of ``{"key1": "value1", "key2": "value2"}`` and should be serialized before being passed, for example: ``{"domain\\": "\\www.test.com\\"}``.

For a clear and intuitive understanding of the specific functions of action parameters, you can see the console.

This API returns the following:

```
{
  "Response": {
    "RequestId": "3e7fa74e-9045-4f01-88d4-ee158affe905",
    Common: [
      # General parameters,
      corresponding to TaskActionGeneralConfiguration in the subsequent action
      experiment creation.
      {
        "ActionId": 466,
        "ActionName": DNS tampering,
        "ConfigDetail": [
          {
            "Type": "input",
            "Label": Action Alias,
            "Field": AliasTitle,
            # Action parameter
            key
            "DefaultValue": "",
            # The default
            value for the action parameter
            "Config": "{}",
            "Required": 0,
            # If it is
            required (0 -- No 1 -- Yes)
            "Validate": "{}",
            "Visible": "{}"
          },
          {
            "Type": "number",
            "Label": Pre-Wait Time (s),
            "Field": "PreTimeWait",
            "DefaultValue": "0",
```

```

        Config: "{ \"max\": 86400, \"min\": 0,
\"tooltip\": \"Only for auto-advance mode\"}",
        "Required": 0,
        "Validate": "{}",
        "Visible": "{}"
    },
    {
        "Type": "number",
        "Label": "Post-Wait Time (s)",
        "Field": "AfterTimeWait",
        "DefaultValue": "0",
        Config: "{ \"max\": 86400, \"min\": 0,
\"tooltip\": \"Only for auto-advance mode\"}",
        "Required": 0,
        "Validate": "{}",
        "Visible": "{}"
    },
    {
        "Type": "number",
        "Label": "Action Timeout Period (s)",
        "Field": "ActionTimeout",
        "DefaultValue": "1800",
        "Config": "{ \"max\": 86400, \"min\": 0,
\"tooltip\": \"Action timeout period\"}",
        "Required": 0,
        "Validate": "{}",
        "Visible": "{ \"op\": \"<\", \"type\":
\"need_insert\", \"value\": 0, \"relatedField\": \"ActionTimeout\"}"
    }
]
}
],
Results: [
    # Custom parameters, corresponding
to TaskActionCustomConfiguration in the subsequent action experiment
creation.
    {
        "ActionId": 466,
        "ActionName": "DNS tampering",
        "ConfigDetail": [
            {

```

```

        "Type": "number",
        "Label": Duration (s),
        "Field": "duration",
        "DefaultValue": "180",
        "Config": "{ \"max\": 1800, \"min\": 0 }",
        "Required": 1,
        "Validate": "{}",
        "Visible": "{}"
    },
    {
        "Type": "input",
        "Label": Domain Name,
        "Field": "domain",           # Action parameter
key
        "DefaultValue": "",         # The default
value for the action parameter
        "Config": "{}",
        "Required": 1,             # If it is required
(0 -- No 1 -- Yes)
        "Validate": "{}",
        "Visible": "{}"
    },
    {
        "Type": "input",
        "Label": "IP",
        "Field": "ip",
        "DefaultValue": "",
        "Config": "{}",
        "Required": 1,
        "Validate": "{}",
        "Visible": "{}"
    }
]
}
],
"ResourceOffline": []
}
}

```

## API Request

Refer to API description in [Creating an Experiment via Action](#), and make the following request:

### Note:

For container-type resource objects, a unique instance is identified by the combination of {ClusterId} + {NodeName} + {NameSpace} + {PodName}. When the parameter TaskInstances is passed in, this map needs to be serialized.

For example: `{"ClusterId":"cls-xxxx","PodName":"pod-xxxxxx","NodeName":"xxxxxxxx","NameSpace":"default-xxxxxx"}`

```
POST / HTTP/1.1
Host: cfg.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: CreateTaskFromTemplate
<Common request parameters>

{
  "TaskActionId": 462, #
  Action ID
  TaskInstances: ["ins-xxxxxxxx"], #
  Resource object instance ID
  "TaskTitle": Network Packet Loss,
  # Experiment name
  TaskDescription: "This experiment was created from the openapi",
  # Experiment description
  TaskActionCustomConfiguration: "{\"interfaces\": \"eth0\"}" #
  Custom action parameters, should be serialized.
}
```

## API Output

```
{
  "Response": {
    "RequestId": "f0aee8ac-2ed3-4a7f-a25b-f0d7d228dd30",
    "TaskId": 150
  }
}
```

At this point, you can click **Experiment management** in the console to view the created experiment, or you can inquire about it through the API.

**Note:**

If you need to delete a created experiment, see the [Delete Experiments](#) API description.

## Query the Experiment

### API Request

Refer to API description in [Query Experiments](#), and make the following request:

```
POST / HTTP/1.1
Host: cfg.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: DescribeTask
<Common request parameters>

{
  "RequestId": "02185fc4-0e8f-49ed-a8d5-6d0788d0e60c",
  "TaskId": "3256" # The experiment
ID returned from the experiment creation process mentioned above
}
```

### API Output

```
{
  "RequestId": "02185fc4-0e8f-49ed-a8d5-6d0788d0e60c",
  "Task": {
    "TaskId": 3256,
    "TaskTitle": "This is an example of creating an experiment from
the API",
    "TaskDescription": "Test an empty operation action",
    "TaskTag": "",
    "TaskStatus": 1002,
    "TaskStatusType": 0,
    "TaskProtectStrategy": null,
    "TaskCreateTime": "2023-08-14 11:55:02",
    "TaskUpdateTime": "2023-08-14 14:48:00",
```

```
"TaskStartTime": "2023-08-14 14:48:01",
"TaskEndTime": null,
"TaskExpect": null,
"TaskSummary": null,
"TaskMode": 1,
"TaskRegionId": 1,
"TaskPauseDuration": 60,
"TaskOwnerUin": "100032429988",
"TaskPlanId": null,
"TaskPlanTitle": null,
"TaskGroups": [
  {
    "TaskGroupActions": [
      {
        "TaskGroupInstances": [
          {
            "TaskGroupInstanceId": 24375,
            # Task action instance ID
            "TaskGroupInstanceObjectId": "ins-bfydnv
            ta", # Resource object ID
            "TaskGroupInstanceStatus": 3001,
            "TaskGroupInstanceStatusType": 0,
            "TaskGroupInstanceExecuteLog": null,
            "TaskGroupInstanceStartTime": null,
            "TaskGroupInstanceEndTime": null,
            "TaskGroupInstanceCreateTime": "2023-08-
            14 14:48:00",
            "TaskGroupInstanceUpdateTime": "2023-08-
            14 14:48:00",
            "TaskGroupInstanceIsRedo": false,
            "TaskGroupInstanceExecuteTime": null
          },
          {
            "TaskGroupInstanceId": 24376,
            # Task action instance ID
            "TaskGroupInstanceObjectId": "ins-ehxmry
            76", # Resource object ID
            "TaskGroupInstanceStatus": 3001,
            "TaskGroupInstanceStatusType": 0,
            "TaskGroupInstanceExecuteLog": null,
```

```
        "TaskGroupInstanceStartTime": null,
        "TaskGroupInstanceEndTime": null,
        "TaskGroupInstanceCreateTime": "2023-08-
14 14:48:00",
        "TaskGroupInstanceUpdateTime": "2023-08-
14 14:48:00",
        "TaskGroupInstanceIsRedo": false,
        "TaskGroupInstanceExecuteTime": null
    }
},
"TaskGroupActionId": 11395,
# Task action ID
"ActionId": 12,
"ActionTitle": "Empty Operation",
"ActionApiType": 1,
"ActionType": "Platform",
"ActionRisk": "Low risk",
"ActionAttribute": 1,
"TaskGroupActionOrder": 1,
"TaskGroupActionGeneralConfiguration": "{\"Alias
Title\": \"\", \"PreTimeWait\": 0, \"ActionTimeout\": 1800, \"AfterTimeW
ait\": 0}\",
"TaskGroupActionCustomConfiguration": "{}",
"TaskGroupActionStatus": 2002,
"TaskGroupActionStatusType": 0,
"TaskGroupActionRandomId": 156878,
"TaskGroupActionRecoverId": 193278,
"TaskGroupActionExecuteId": null,
"TaskGroupActionCreateTime": "2023-08-14 11:55:0
2",
"TaskGroupActionUpdateTime": "2023-08-14 14:48:0
0",
"IsExecuteRedo": false,
"TaskGroupActionExecuteTime": null
},
{
"TaskGroupInstances": [
{
"TaskGroupInstanceId": 24377,
# Task action instance ID
```

```
    "TaskGroupInstanceObjectId": "ins-bfydnv
ta",    # Resource object ID
    "TaskGroupInstanceStatus": 3001,
    "TaskGroupInstanceStatusType": 0,
    "TaskGroupInstanceExecuteLog": null,
    "TaskGroupInstanceStartTime": null,
    "TaskGroupInstanceEndTime": null,
    "TaskGroupInstanceCreateTime": "2023-08-
14 14:48:00",
    "TaskGroupInstanceUpdateTime": "2023-08-
14 14:48:00",
    "TaskGroupInstanceIsRedo": false,
    "TaskGroupInstanceExecuteTime": null
  },
  {
    "TaskGroupInstanceId": 24378,
# Task action instance ID
    "TaskGroupInstanceObjectId": "ins-ehxmry
76",    # Resource object ID
    "TaskGroupInstanceStatus": 3001,
    "TaskGroupInstanceStatusType": 0,
    "TaskGroupInstanceExecuteLog": null,
    "TaskGroupInstanceStartTime": null,
    "TaskGroupInstanceEndTime": null,
    "TaskGroupInstanceCreateTime": "2023-08-
14 14:48:00",
    "TaskGroupInstanceUpdateTime": "2023-08-
14 14:48:00",
    "TaskGroupInstanceIsRedo": false,
    "TaskGroupInstanceExecuteTime": null
  }
],
"TaskGroupActionId": 11396,
# Task action ID
  "ActionId": 13,
  "ActionTitle": "Empty Operation (rollback)",
  "ActionApiType": 1,
  "ActionType": "Platform",
  "ActionRisk": "Low risk",
  "ActionAttribute": 2,
```

```
        "TaskGroupActionOrder": 2,
        "TaskGroupActionGeneralConfiguration": "{\"PreTimeWait\": 0, \"ActionTimeout\": 1800, \"AfterTimeWait\": 0}\",
        "TaskGroupActionCustomConfiguration": "{}",
        "TaskGroupActionStatus": 2001,
        "TaskGroupActionStatusType": 0,
        "TaskGroupActionRandomId": 193278,
        "TaskGroupActionRecoverId": null,
        "TaskGroupActionExecuteId": 156878,
        "TaskGroupActionCreateTime": "2023-08-14 11:55:02",
        "TaskGroupActionUpdateTime": "2023-08-14 11:55:02",
        "IsExecuteRedo": false,
        "TaskGroupActionExecuteTime": null
    }
],
"TaskGroupId": 4684,
# Action group ID
"TaskGroupTitle": "abc",
"TaskGroupDescription": "abc",
"TaskGroupOrder": 1,
"TaskGroupMode": 1,
"TaskGroupInstanceList": [
    "ins-bfydnvta",
    "ins-ehxmry76"
],
"ObjectTypeid": 1,
"TaskGroupCreateTime": "2023-08-14 11:55:02",
"TaskGroupUpdateTime": "2023-08-14 11:55:02",
"TaskGroupInstancesExecuteRule": [
    {
        "TaskGroupInstancesExecuteMode": 1
    }
],
"TaskGroupSelectedInstanceList": [
    "ins-bfydnvta",
    "ins-ehxmry76"
],
"TaskGroupDiscardInstanceList": []
```

```
    },
  ],
  "TaskMonitors": [],
  "TaskPolicy": null,
  "Tags": []
},
"ReportInfo": null
}
```

## Execute the Experiment

### API Request

Refer to API description in [Execute Experiments](#), and make the following request:

```
POST / HTTP/1.1
Host: cfg.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: ExecuteTask
<Common request parameters>

{
  "TaskId": "3256"
}
```

### API Output

```
{
  "Response": {
    "RequestId": "46924e75-a149-4130-aac0-853dbf0abea9"
  }
}
```

## Execute Action

### API Request

Refer to API description in [Execute Action](#), and make the following request:

```
POST / HTTP/1.1
```

```
Host: cfg.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: ExecuteTaskInstance
<Common request parameters>

{
  "TaskId": "3256",
  "TaskActionId": "11396", # Task action ID
  (obtained from the experiment inquiry response)
  "TaskInstanceIds": [
    "xxxxxxxx-01", # Task action in
  stance ID (obtained from the experiment inquiry response)
    "xxxxxxxx-02"
  ],
  "IsOperateAll": true, # Whether to execute the entire task. When true is set, TaskInstanceIds will be ignored, and all instances provided during experiment creation will be executed.
  "ActionType": 2, # 2--Execute, 3--Skip, 5--Retry
  "TaskGroupId": 4684, # Action group ID
  (obtained from the experiment inquiry response)
}
```

## API Output

```
{
  "Response": {
    "RequestId": "6549ed1a-911f-46dd-b6cd-2c02d5bd180f"
  }
}
```

### Note:

The action execution here supports skip and retry operations, which can be controlled by adjusting the value of `ActionType`:

- 3: Skip
- 5: Retry

## End the Experiment

## API Request

Refer to API description in [End Experiment](#), and make the following request:

```
POST / HTTP/1.1
Host: cfg.tencentcloudapi.com
Content-Type: application/json
X-TC-Action: ModifyTaskRunStatus
<Common request parameters>

{
  "TaskId": 3256,           # Experiment task ID
  "Status": 1004,         # End status code, no need to modify
  "Summary": "This experiment meets the expectations",
  # Experiment conclusion
  "IsExpect": true,       # Is the execution result as expected?
}
```

## API Output

```
{
  "Response": {
    "RequestId": "e38eca72-e4ae-4a86-9696-7df399e672bd"
  }
}
```

### Note:

To view the logs of the experiment, refer to API description in [Getting Experiments Process Log](#).