Anti-DDoS Service

API Reference

Issue 08

Date 2025-11-26

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

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

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

Contents

1 Before You Start

Huawei Cloud provides multiple security solutions to defend against DDoS attacks. You can select an appropriate one based on your service requirements. Huawei Cloud Anti-DDoS Service provides three sub-services: Cloud Native Anti-DDoS Basic, Cloud Native Anti-DDoS Advanced, and Advanced Anti-DDoS.

This document describes how to make application programming interface (API) calls to use the AAD, such as querying or updating Anti-DDoS defense policies. For details about all supported operations, see **API Overview**.

Endpoints

An endpoint is the request address for calling an API. Endpoints vary depending on services and regions. For the endpoints of all services, see **Anti-DDoS Endpoints** and **AAD Endpoints**.

Concepts

- Account

  An account is created upon successful registration. The account has full access permissions for all of its cloud services and resources. It can be used to reset user passwords and grant user permissions. The account is a payment entity and should not be used to perform routine management. For security purposes, create IAM users and grant them permissions for routine management.

- User

  An IAM user is created by an account in IAM to use cloud services. Each IAM user has its own identity credentials (password and access keys).

  An IAM user can view the account ID and user ID on the **My Credentials** page of the management console. The domain name, username, and password will be required for API authentication.

- Region

  Regions are divided based on geographical location and network latency. Public services, such as Elastic Cloud Server (ECS), Elastic Volume Service (EVS), Object Storage Service (OBS), Virtual Private Cloud (VPC), Elastic IP (EIP), and Image Management Service (IMS), are shared within the same region. Regions are classified as universal regions and dedicated regions. A

universal region provides universal cloud services for common tenants. A dedicated region provides services of the same type only or for specific tenants.

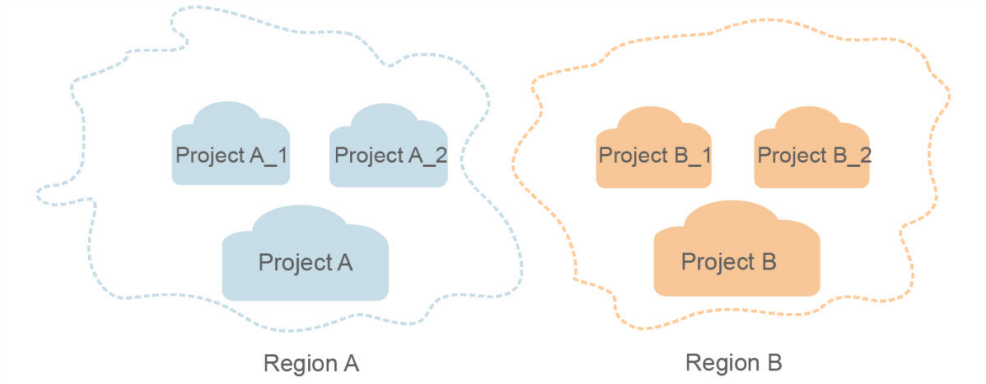
- Availability Zone (AZ)

  An AZ comprises one or multiple physical data centers equipped with independent ventilation, fire, water, and electricity facilities. Compute, network, storage, and other resources in an AZ are logically divided into multiple clusters. AZs within a region are interconnected using high-speed optical fibers to support cross-AZ high-availability systems.

- Project

  Projects group and isolate resources (including compute, storage, and network resources) across physical regions. A default project is provided for each region, and subprojects can be created under each default project. Users can be granted permissions to access all resources in a specific project. For more refined access control, create subprojects under a project and create resources in the subprojects. Users can then be assigned permissions to access only specific resources in the subprojects.

  Figure 1-1 Project isolation model

- Enterprise Project

  Enterprise projects group and logically isolate resources. An enterprise project can contain resources of multiple regions, and resources can be added to or removed from the enterprise project.

2 API Overview

You can use all functions of Anti-DDoS through its APIs.

Type	API Type	Description
CNAD Basic	Tasks	Query Anti-DDoS tasks.
	Protection Management	Query protection status and update protection policies.
	Alarm Configuration Management	Query and update alarm configuration information.
	Default Protection Policy Management	Query, configure, and delete default protection policies.

3 API Calling

3.1 Making an API Request

This section describes the structure of a RESTful API request, and uses the IAM API for **Creating an IAM User** as an example to describe how to call an API.

Request URI

A request URI is in the following format:

{URI-scheme} :// {Endpoint} / {resource-path} ? {query-string}

Although a request URI is included in the request header, most programming languages or frameworks require the request URI to be transmitted separately.

- **URI-scheme**:

  Protocol used to transmit requests. All APIs use HTTPS.

- **Endpoint**:

  Domain name or IP address of the server bearing the REST service. The endpoint varies between services in different regions. It can be obtained from **Regions and Endpoints**.

  For example, the endpoint of IAM in region **CN-Hong Kong** is **iam.ap-southeast-1.myhuaweicloud.com**.

- **resource-path**:

  Access path of an API for performing a specified operation. Obtain the path from the URI of an API. For example, the resource-path of the API for **Creating an IAM User** is **/v3.0/OS-USER/users**.

- **query-string**:

  Query parameter, which is optional. Ensure that a question mark (?) is included before each query parameter that is in the format of "Parameter name=Parameter value". For example, **?limit=10** indicates that a maximum of 10 data records will be displayed.

For example, if you want to create an IAM user, use the endpoint of any region (for example, the endpoint of **CN-Hong Kong**: **iam.ap-southeast-1.myhuaweicloud.com**) and the resource-path (**/v3.0/OS-USER/users**)

in the URI of the API for **creating an IAM User**. Then, construct the URI as follows:

```
https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
```

Figure 3-1 Example URI

NOTE

To simplify the URI display in this document, each API is provided only with a **resource-path** and a request method. The **URI-scheme** of all APIs is **HTTPS**, and the endpoints of all APIs in the same region are identical.

Request Methods

The HTTP protocol defines the following request methods that can be used to send a request to the server:

- **GET**: requests the server to return specified resources.
- **PUT**: requests the server to update specified resources.
- **POST**: requests the server to add resources or perform special operations.
- **DELETE**: requests the server to delete specified resources, for example, an object.
- **HEAD**: same as GET except that the server must return only the response header.
- **PATCH**: requests the server to update partial content of a specified resource. If the resource does not exist, a new resource will be created.

For example, in the URI of the API for **Creating an IAM User**, the request method is **POST**. The request is as follows:

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
```

Request Header

You can also add additional header fields to a request, such as the fields required by a specified URI or HTTP method. For example, to request for the authentication information, add **Content-Type**, which specifies the request body type.

Common request header fields are as follows:

- **Content-Type**: specifies the request body type or format. This field is mandatory and its default value is **application/json**. Other values of this field will be provided for specific APIs if any.
- **Authorization**: specifies signature authentication information. This field is optional. When AK/SK authentication is enabled, this field is automatically specified when SDK is used to sign the request. For more details, see **AK/SK-based Authentication**.

- **X-Sdk-Date**: specifies the time when a request is sent. This field is optional. When AK/SK authentication is enabled, this field is automatically specified when SDK is used to sign the request. For more details, see **AK/SK-based Authentication**.
- **X-Auth-Token**: specifies a user token only for token-based API authentication. The user token is a response to the API used to **obtain a user token**. This API is the only one that does not require authentication.
- **X-Project-ID**: specifies subproject ID. This field is optional and can be used in multi-project scenarios. The **X-Project-ID** field is mandatory in the request header for accessing resources in a subproject through AK/SK-based authentication.
- **X-Domain-ID**: account ID, which is optional. When you call APIs of global services using AK/SK-based authentication, **X-Domain-ID** needs to be configured in the request header.

For the API for **Creating an IAM User**, if AK/SK-based authentication is enabled, the request with the header is as follows:

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
Content-Type: application/json
X-Sdk-Date: 20240416T095341Z
Authorization: SDK-HMAC-SHA256 Access=**********, SignedHeaders=content-type;host;x-sdk-date,
Signature=**********
```

Request Body

The body of a request is often sent in a structured format as specified in **Content-Type**. The request body transfers content except the request header. The request body transfers content other than the request header. If the request body contains Chinese characters, set Content-type to utf-8, for example, **Content-Type: application/json; charset=utf-8**.

The request body varies between APIs. Some APIs do not require the request body, such as the APIs requested using the GET and DELETE methods.

The following shows an example request (a request body included) of the API for **Creating an IAM User**. You can learn about request parameters and related description from this example. The bold parameters need to be replaced for a real request.

- **accountid**: ID of the account to which the IAM user belongs.
- **username**: IAM username to be created.
- **email**: email address of the IAM user.
- **************: password of the IAM user.

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
Content-Type: application/json
X-Sdk-Date: 20240416T095341Z
Authorization: SDK-HMAC-SHA256 Access=**********, SignedHeaders=content-type;host;x-sdk-date,
Signature=**********

{
    "user": {
        "domain_id": "accountid",
        "name": "username",
        "password": "**********",
        "email": "email",
        "description": "IAM User Description"
```



```
  }
}
```

By now, all data required for an API request is available. You can send the request to call the API through curl, Postman, or coding.

3.2 Authentication

Requests for calling an API can be authenticated using either of the following methods:

- AK/SK-based authentication: Requests are authenticated by encrypting the request body using an AK/SK pair.
- Token-based authentication: Requests are authenticated using a token.

AK/SK-based Authentication

NOTE

- AK/SK-based authentication supports API requests with a body not larger than 12 MB. For API requests with a larger body, token-based authentication is recommended.
- You can use the AK/SK in a permanent or temporary access key. The **X-Security-Token** field must be configured if the AK/SK in a temporary access key is used, and the field value is **security_token** of the temporary access key.

In AK/SK-based authentication, AK/SK is used to sign requests and the signature is then added to the requests for authentication.

- AK: access key ID, which is a unique identifier used in conjunction with a secret access key to sign requests cryptographically.
- SK: secret access key used in conjunction with an AK to sign requests cryptographically. It identifies a request sender and prevents the request from being modified.

In AK/SK-based authentication, you can use an AK/SK to sign requests based on the signature algorithm or use the signing SDK to sign requests. For details about how to sign requests and use the signing SDK, see **AK/SK Signing and Authentication Guide**.

NOTICE

The signing SDK is only used for signing requests and is different from the SDKs provided by services.

Token-based Authentication

NOTE

- The validity period of a token is 24 hours. When using a token for authentication, cache it to prevent frequently calling the IAM API used to obtain a user token.
- Ensure that the token is valid when you use it. Using a token that will soon expire may cause API calling failures.

A token specifies temporary permissions in a computer system. During API authentication using a token, the token is added to requests to get permissions for calling the API.

The token can be obtained by calling the required API. For more information, see **Obtaining a User Token**. A project-level token is required for calling this API, that is, **auth.scope** must be set to **project** in the request body. Example:

```
{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "username",
                    "password": "********",
                    "domain": {
                        "name": "domainname"
                    }
                }
            }
        },
        "scope": {
            "project": {
                "name": "xxxxxxxx"
            }
        }
    }
}
```

After a token is obtained, the **X-Auth-Token** header field must be added to requests to specify the token when calling other APIs. For example, if the token is **ABCDEFG....**, add **X-Auth-Token: ABCDEFG....** to a request as follows:

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
Content-Type: application/json
X-Auth-Token: ABCDEFG....
```

3.3 Response

Status Code

After sending a request, you will receive a response, including a status code, response header, and response body.

A status code is a group of digits, ranging from 1xx to 5xx. It indicates the status of a request. For more information, see **HTTP Status Codes of Cloud Native Anti-DDoS Basic**.

If status code **201** is returned for the API for **Creating an IAM User**, the request is successful.

Response Header

Similar to a request, a response also has a header, for example, **Content-Type**.

For the API for **Creating an IAM User**, the message header shown in **Figure 3-2** is returned.

Figure 3-2 Response header fields for the API used to create an IAM user

```
"X-Frame-Options": "SAMEORIGIN",
"X-IAM-ETag-id": "2562365939-d8f6f12921974cb097338ac11fceac8a",
"Transfer-Encoding": "chunked",
"Strict-Transport-Security": "max-age=31536000; includeSubdomains;",
"Server": "api-gateway",
"X-Request-Id": "af2953f2bcc67a42325a69a19e6c32a2",
"X-Content-Type-Options": "nosniff",
"Connection": "keep-alive",
"X-Download-Options": "noopen",
"X-XSS-Protection": "1; mode=block;",
"X-IAM-Trace-Id": "token_[redacted]_null_af2953f2bcc67a42325a69a19e6c32a2",
"Date": "Tue, 21 May 2024 09:03:40 GMT",
"Content-Type": "application/json; charset=utf8"
```

(Optional) Response Body

The body of a response is often returned in structured format as specified in the **Content-Type** header field. The response body transfers content except the response header.

For the API for **Creating an IAM User**, the returned message header shown is as follows. For the sake of space, only part of the content is displayed here.

```
{
    "user": {
        "id": "c131886aec...",
        "name": "IAMUser",
        "description": "IAM User Description",
        "areacode": "",
        "phone": "",
        "email": "***@***.com",
        "status": null,
        "enabled": true,
        "pwd_status": false,
        "access_mode": "default",
        "is_domain_owner": false,
        "xuser_id": "",
        "xuser_type": "",
        "password_expires_at": null,
        "create_time": "2024-05-21T09:03:41.000000",
        "domain_id": "d78cbac1.....",
        "xdomain_id": "30086000.....",
        "xdomain_type": "",
        "default_project_id": null
    }
}
```

If an error occurs during API calling, an error code and a message will be displayed. The following shows an error response body.

```
{
    "error_msg": "Request body is invalid.",
    "error_code": "IAM.0011"
}
```

In the response body, **error_code** is an error code, and **error_msg** provides information about the error.

4 Cloud Native Anti-DDoS Basic APIs

4.1 Anti-DDoS Task Management

4.1.1 Querying Anti-DDoS Tasks

Function

This API enables you to query the execution status of a specified Anti-DDoS configuration task.

Calling Method

For details, see **Calling APIs**.

URI

GET /v2/{project_id}/query-task-status

Table 4-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Table 4-2 Query Parameters

Parameter	Mandatory	Type	Description
task_id	Yes	String	Task ID (non-negative integer) string.

Request Parameters

Table 4-3 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-4 Response body parameters

Parameter	Type	Description
task_status	String	Task status. The options are as follows: • **success**: successful • **failed**: failed • **waiting**: waiting • **running**: running • **preprocess**: preprocessing • **ready**: ready
task_msg	String	Additional information about a task.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.

```
{
  "task_status" : "success",
  "task_msg" : ""
}
```


SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v2.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v2.*;
import com.huaweicloud.sdk.antiddos.v2.model.*;


public class ShowNewTaskStatusSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowNewTaskStatusRequest request = new ShowNewTaskStatusRequest();
        try {
            ShowNewTaskStatusResponse response = client.showNewTaskStatus(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v2.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v2 import *

if __name__ == "__main__":
```



```
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowNewTaskStatusRequest()
        response = client.show_new_task_status(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowNewTaskStatusRequest{}
    response, err := client.ShowNewTaskStatus(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```


More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2 DDoS Protection Management

4.2.1 Querying the List of EIP Defense Statuses

Function

Query the defense statuses of all EIPs, regardless of whether an EIP has been bound to an ECS or not.

Calling Method

For details, see **Calling APIs**.

URI

GET /v1/{project_id}/antiddos

Table 4-5 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Table 4-6 Query Parameters

Parameter	Mandatory	Type	Description
status	No	String	Values: • **normal**: normal • **configging**: configuration in progress • **notConfig**: not configured • **packetcleaning**: cleaning • **packetdropping**: blackhole If this parameter is not specified, the defense statuses of all ECSs are displayed in the Neutron-queried order by default.
limit	No	String	Maximum number of returned results. The value ranges from 1 to 100.
offset	No	String	Offset. The value ranges from 0 to 2147483647.
ip	No	String	IP address, which can be in IPv4 or IPv6 format. Partial query is supported. For example, if you enter **?ip=192.168**, the EIP protection status of 192.168.111.1 and 10.192.168.8 will be returned.

Request Parameters

Table 4-7 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-8 Response body parameters

Parameter	Type	Description
total	Integer	Total number of EIPs.
ddosStatus	Array of **DDosStatus** objects	Protection status list.

Table 4-9 DDosStatus

Parameter	Type	Description
floating_ip_id	String	EIP ID.
floating_ip_address	String	Floating IP address.
network_type	String	EIP type. The value can be: • **EIP**: EIP that is bound or not bound with ECS. • **ELB**: EIP that is bound with ELB.
status	String	Protection status. The options are as follows: • **normal**: normal • **configging**: configuration in progress • **notConfig**: not configured • **packetcleaning**: cleaning • **packetdropping**: blackhole
blackhole_endtime	Long	End time of a black hole.
protect_type	String	Protection type.
traffic_threshold	Long	Traffic threshold.
http_threshold	Long	HTTP traffic threshold.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.

```
{
  "total" : 1,
  "ddosStatus" : [ {
    "floating_ip_id" : "18e6ace5-eb36-4196-a15e-1e000c24e026",
    "floating_ip_address" : "139.9.116.167",
    "network_type" : "EIP",
    "status" : "normal",
    "blackhole_endtime" : 0,
    "protect_type" : "default",
    "traffic_threshold" : 99,
    "http_threshold" : 0
  } ]
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v2.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v2.*;
import com.huaweicloud.sdk.antiddos.v2.model.*;


public class ListDDosStatusSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ListDDosStatusRequest request = new ListDDosStatusRequest();
        try {
            ListDDosStatusResponse response = client.listDDosStatus(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
```



```
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v2.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListDDosStatusRequest()
        response = client.list_d_dos_status(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"
```



```
    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListDDosStatusRequest{}
    response, err := client.ListDDosStatus(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2.2 Querying Anti-DDoS Specifications

Function

This API allows you to query available Anti-DDoS defense policies. You can select a policy for Anti-DDoS traffic cleaning.

Calling Method

For details, see **Calling APIs**.

URI

GET /v2/{project_id}/antiddos/query-config-list

Table 4-10 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Request Parameters

Table 4-11 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-12 Response body parameters

Parameter	Type	Description
traffic_limited_list	Array of **TriggerBpsDict** objects	List of traffic limits.
http_limited_list	Array of **TriggerQpsDict** objects	List of HTTP limits.
connection_limited_list	Array of **CleanLimitDict** objects	List of connection limits.
extend_ddos_config	Array of **ExtendDDoSSet** objects	Extended configuration list.

Table 4-13 TriggerBpsDict

Parameter	Type	Description
traffic_pos_id	Long	Position ID of traffic. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
traffic_per_second	Long	Threshold of traffic per second (Mbit/s).
packet_per_second	Long	Threshold of number of packets per second.

Table 4-14 TriggerQpsDict

Parameter	Type	Description
http_request_pos_id	Long	Segment ID of the HTTP request counts. The value is fixed to **1**.
http_packet_per_second	Long	Threshold of number of HTTP requests per second.

Table 4-15 CleanLimitDict

Parameter	Type	Description
cleaning_access_pos_id	Long	Position ID of access limit during cleaning. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
new_connection_limited	Long	Number of new connections of a source IP address.
total_connection_limited	Long	Total number of connections of a source IP address.

Table 4-16 ExtendDDoSSet

Parameter	Type	Description
SetID	Long	Configuration segment ID.
new_connection_limited	Long	Number of new connections of a source IP address.
total_connection_limited	Long	Total number of connections of a source IP address.
http_packet_per_second	Long	Threshold of number of HTTP requests per second.
traffic_per_second	Long	Threshold of traffic per second (Mbit/s).
packet_per_second	Long	Threshold of number of packets per second.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.


```
{
  "traffic_limited_list" : [ {
    "traffic_pos_id" : 1,
    "traffic_per_second" : 10,
    "packet_per_second" : 2000
  }, {
    "traffic_pos_id" : 2,
    "traffic_per_second" : 30,
    "packet_per_second" : 6000
  }, {
    "traffic_pos_id" : 3,
    "traffic_per_second" : 50,
    "packet_per_second" : 10000
  }, {
    "traffic_pos_id" : 4,
    "traffic_per_second" : 70,
    "packet_per_second" : 15000
  }, {
    "traffic_pos_id" : 5,
    "traffic_per_second" : 100,
    "packet_per_second" : 20000
  }, {
    "traffic_pos_id" : 6,
    "traffic_per_second" : 150,
    "packet_per_second" : 25000
  }, {
    "traffic_pos_id" : 7,
    "traffic_per_second" : 200,
    "packet_per_second" : 35000
  }, {
```



```
  "traffic_pos_id" : 8,
  "traffic_per_second" : 250,
  "packet_per_second" : 50000
}, {
  "traffic_pos_id" : 9,
  "traffic_per_second" : 300,
  "packet_per_second" : 70000
}, {
  "traffic_pos_id" : 88,
  "traffic_per_second" : 1000,
  "packet_per_second" : 300000
} ],
"http_limited_list" : [ {
  "http_request_pos_id" : 1,
  "http_packet_per_second" : 100
}, {
  "http_request_pos_id" : 2,
  "http_packet_per_second" : 150
}, {
  "http_request_pos_id" : 3,
  "http_packet_per_second" : 240
}, {
  "http_request_pos_id" : 4,
  "http_packet_per_second" : 350
}, {
  "http_request_pos_id" : 5,
  "http_packet_per_second" : 480
}, {
  "http_request_pos_id" : 6,
  "http_packet_per_second" : 550
}, {
  "http_request_pos_id" : 7,
  "http_packet_per_second" : 700
}, {
  "http_request_pos_id" : 8,
  "http_packet_per_second" : 850
}, {
  "http_request_pos_id" : 9,
  "http_packet_per_second" : 1000
}, {
  "http_request_pos_id" : 10,
  "http_packet_per_second" : 1500
}, {
  "http_request_pos_id" : 11,
  "http_packet_per_second" : 2000
}, {
  "http_request_pos_id" : 12,
  "http_packet_per_second" : 3000
}, {
  "http_request_pos_id" : 13,
  "http_packet_per_second" : 5000
}, {
  "http_request_pos_id" : 14,
  "http_packet_per_second" : 10000
}, {
  "http_request_pos_id" : 15,
  "http_packet_per_second" : 20000
} ],
"connection_limited_list" : [ {
  "cleaning_access_pos_id" : 1,
  "new_connection_limited" : 10,
  "total_connection_limited" : 30
}, {
  "cleaning_access_pos_id" : 2,
  "new_connection_limited" : 20,
  "total_connection_limited" : 100
}, {
  "cleaning_access_pos_id" : 3,
  "new_connection_limited" : 30,
```



```
    "total_connection_limited" : 200
  }, {
    "cleaning_access_pos_id" : 4,
    "new_connection_limited" : 40,
    "total_connection_limited" : 250
  }, {
    "cleaning_access_pos_id" : 5,
    "new_connection_limited" : 50,
    "total_connection_limited" : 300
  }, {
    "cleaning_access_pos_id" : 6,
    "new_connection_limited" : 60,
    "total_connection_limited" : 500
  }, {
    "cleaning_access_pos_id" : 7,
    "new_connection_limited" : 70,
    "total_connection_limited" : 600
  }, {
    "cleaning_access_pos_id" : 8,
    "new_connection_limited" : 80,
    "total_connection_limited" : 700
  } ],
  "extend_ddos_config" : [ ]
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v2.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v2.*;
import com.huaweicloud.sdk.antiddos.v2.model.*;


public class ListNewConfigsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ListNewConfigsRequest request = new ListNewConfigsRequest();
        try {
            ListNewConfigsResponse response = client.listNewConfigs(request);
            System.out.println(response.toString());
```



```
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v2.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListNewConfigsRequest()
        response = client.list_new_configs(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
```



```
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListNewConfigsRequest{}
    response, err := client.ListNewConfigs(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2.3 Querying Weekly Defense Statistics

Function

This API allows you to query weekly defense statistics about all your EIPs, including the number of blocked DDoS attacks, number of attacks, and ranking by the number of attacks. Currently, you can query weekly statistics up to four weeks before the current time. Data older than four weeks cannot be queried.

Calling Method

For details, see **Calling APIs**.

URI

GET /v1/{project_id}/antiddos/weekly

Table 4-17 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Table 4-18 Query Parameters

Parameter	Mandatory	Type	Description
period_start_date	No	String	Start date of a week.

Request Parameters

Table 4-19 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-20 Response body parameters

Parameter	Type	Description
ddos_intercept_times	Integer	Number of DDoS attacks blocked in a week.
weekdata	Array of **WeeklyCount** objects	Number of attacks in a week.
top10	Array of **WeeklyTop10** objects	Top 10 attacked IP addresses.

Table 4-21 WeeklyCount

Parameter	Type	Description
ddos_intercept_times	Integer	Number of DDoS attacks blocked.
ddos_blackhole_times	Integer	Number of DDoS black holes.
max_attack_bps	Integer	Maximum attack traffic.
max_attack_conns	Integer	Maximum number of attack connections.
period_start_date	Long	Start time.

Table 4-22 WeeklyTop10

Parameter	Type	Description
floating_ip_address	String	EIP.
times	Integer	Number of DDoS attacks blocked by cleaning operations and black holes.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.


```
{
  "ddos_intercept_times" : 0,
  "weekdata" : [ {
    "ddos_intercept_times" : 0,
    "ddos_blackhole_times" : 0,
    "max_attack_bps" : 0,
    "max_attack_conns" : 0,
    "period_start_date" : 1605496722606
  }, {
    "ddos_intercept_times" : 0,
    "ddos_blackhole_times" : 0,
    "max_attack_bps" : 0,
    "max_attack_conns" : 0,
    "period_start_date" : 1605583122606
  }, {
    "ddos_intercept_times" : 0,
    "ddos_blackhole_times" : 0,
    "max_attack_bps" : 0,
    "max_attack_conns" : 0,
    "period_start_date" : 1605669522606
  }, {
```



```
    "ddos_intercept_times" : 0,
    "ddos_blackhole_times" : 0,
    "max_attack_bps" : 0,
    "max_attack_conns" : 0,
    "period_start_date" : 1605755922606
  }, {
    "ddos_intercept_times" : 0,
    "ddos_blackhole_times" : 0,
    "max_attack_bps" : 0,
    "max_attack_conns" : 0,
    "period_start_date" : 1605842322606
  }, {
    "ddos_intercept_times" : 0,
    "ddos_blackhole_times" : 0,
    "max_attack_bps" : 0,
    "max_attack_conns" : 0,
    "period_start_date" : 1605928722606
  }, {
    "ddos_intercept_times" : 0,
    "ddos_blackhole_times" : 0,
    "max_attack_bps" : 0,
    "max_attack_conns" : 0,
    "period_start_date" : 1606015122606
  } ],
  "top10" : [ ]
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class ListWeeklyReportsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ListWeeklyReportsRequest request = new ListWeeklyReportsRequest();
        try {
```



```
            ListWeeklyReportsResponse response = client.listWeeklyReports(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListWeeklyReportsRequest()
        response = client.list_weekly_reports(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
```



```
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListWeeklyReportsRequest{}
    response, err := client.ListWeeklyReports(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2.4 Querying Anti-DDoS Policies

Function

This API enables you to query configured Anti-DDoS defense policies. You can query the policy of a specified EIP.

Calling Method

For details, see **Calling APIs**.

URI

GET /v1/{project_id}/antiddos/{floating_ip_id}

Table 4-23 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.
floating_ip_id	Yes	String	ID corresponding to the EIP of a user.

Table 4-24 Query Parameters

Parameter	Mandatory	Type	Description
ip	No	String	EIP of a user.

Request Parameters

Table 4-25 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-26 Response body parameters

Parameter	Type	Description
enable_L7	Boolean	Whether to enable layer-7 protection. The value is fixed to **false**.
traffic_pos_id	Long	Position ID of traffic. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection

Parameter	Type	Description
http_request_pos_id	Long	Segment ID of the HTTP request counts. The value is fixed to **1**.
cleaning_access_pos_id	Long	Position ID of access limit during cleaning. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
app_type_id	Long	Application type ID. The value is fixed to **0**.
antiddos_config_name	String	Protection level name.
antiddos_config_id	String	Protection level ID.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.

```
{
  "enable_L7" : false,
  "traffic_pos_id" : 8,
  "http_request_pos_id" : 8,
  "cleaning_access_pos_id" : 8,
  "app_type_id" : 1,
  "antiddos_config_name" : "test",
  "antiddos_config_id" : "1234567890"
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
```



```
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class ShowDDosSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowDDosRequest request = new ShowDDosRequest();
        request.withFloatingIpId("{floating_ip_id}");
        try {
            ShowDDosResponse response = client.showDDos(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()
```



```
    try:
        request = ShowDDosRequest()
        request.floating_ip_id = "{floating_ip_id}"
        response = client.show_d_dos(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowDDosRequest{}
    request.FloatingIpId = "{floating_ip_id}"
    response, err := client.ShowDDos(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2.5 Updating the Anti-DDoS Protection

Function

This API enables you to update the Anti-DDoS defense policy of a specified EIP. If this API is successfully called, it indicates that the service node receives the request for updating the Anti-DDoS policy of the EIP. To check whether the operation is successful, call the task query API to query the task execution status. For details, see "Querying Anti-DDoS Tasks".

Calling Method

For details, see **Calling APIs**.

URI

PUT /v1/{project_id}/antiddos/{floating_ip_id}

Table 4-27 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.
floating_ip_id	Yes	String	ID corresponding to the EIP of a user.

Table 4-28 Query Parameters

Parameter	Mandatory	Type	Description
ip	No	String	ip

Request Parameters

Table 4-29 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Table 4-30 Request body parameters

Parameter	Mandatory	Type	Description
app_type_id	Yes	Integer	Application type ID. The value is fixed to **0**.
cleaning_access_pos_id	Yes	Integer	Position ID of access limit during cleaning. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
enable_L7	Yes	Boolean	Whether to enable layer-7 protection. The value is fixed to **false**.
http_request_pos_id	Yes	Integer	Segment ID of the HTTP request counts. The value is fixed to **1**.
traffic_pos_id	Yes	Integer	Position ID of traffic. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
antiddos_config_id	No	String	Protection level ID.

Response Parameters

Status code: 200

Table 4-31 Response body parameters

Parameter	Type	Description
error_code	String	Internal error code.
error_msg	String	Internal error description.
task_id	String	ID of a task. This ID can be used to query the status of the task. This field is reserved for future task audit extension and is not required currently.

Example Requests

Update the Anti-DDoS policy for a specified EIP. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
PUT https://{endpoint}/v1/{project_id}/antiddos/{floating_ip_id}

{
  "app_type_id" : 0,
  "cleaning_access_pos_id" : 8,
  "enable_L7" : false,
  "http_request_pos_id" : 1,
  "traffic_pos_id" : 1,
  "antiddos_config_name" : "test",
  "antiddos_config_id" : "1234567890"
}
```

Example Responses

Status code: 200

Request succeeded.

```
{
  "error_code" : "10000000",
  "error_msg" : "The task has been received and is being handled",
  "task_id" : "59385d2a-6266-4d3a-9122-a228c530f557"
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

Update the Anti-DDoS policy for a specified EIP. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.


```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class UpdateDDosSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateDDosRequest request = new UpdateDDosRequest();
        request.withFloatingIpId("{floating_ip_id}");
        UpdateAntiDDosServiceRequestBody body = new UpdateAntiDDosServiceRequestBody();
        body.withTrafficPosId(1);
        body.withHttpRequestPosId(1);
        body.withEnableL7(false);
        body.withCleaningAccessPosId(8);
        body.withAppTypeId(UpdateAntiDDosServiceRequestBody.AppTypeIdEnum.NUMBER_0);
        request.withBody(body);
        try {
            UpdateDDosResponse response = client.updateDDos(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

Update the Anti-DDoS policy for a specified EIP. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
```



```
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = UpdateDDosRequest()
        request.floating_ip_id = "{floating_ip_id}"
        request.body = UpdateAntiDDosServiceRequestBody(
            traffic_pos_id=1,
            http_request_pos_id=1,
            enable_l7=False,
            cleaning_access_pos_id=8,
            app_type_id=0
        )
        response = client.update_d_dos(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

Update the Anti-DDoS policy for a specified EIP. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()
```



```
    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateDDosRequest{}
    request.FloatingIpId = "{floating_ip_id}"
    request.Body = &model.UpdateAntiDDosServiceRequestBody{
        TrafficPosId: int32(1),
        HttpRequestPosId: int32(1),
        EnableL7: false,
        CleaningAccessPosId: int32(8),
        AppTypeId: model.GetUpdateAntiDDosServiceRequestBodyAppTypeIdEnum().E_0,
    }
    response, err := client.UpdateDDos(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2.6 Querying the Traffic of a Specified EIP

Function

You can query the traffic of a specified EIP in the last 24 hours. Traffic is detected at five-minute intervals.

Calling Method

For details, see **Calling APIs**.

URI

GET /v1/{project_id}/antiddos/{floating_ip_id}/daily

Table 4-32 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.
floating_ip_id	Yes	String	ID corresponding to the EIP of a user.

Table 4-33 Query Parameters

Parameter	Mandatory	Type	Description
ip	No	String	EIP of a user.

Request Parameters

Table 4-34 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-35 Response body parameters

Parameter	Type	Description
data	Array of **DailyData** objects	Traffic in the last 24 hours.

Table 4-36 DailyData

Parameter	Type	Description
period_start	Long	Start time.

Parameter	Type	Description
bps_in	Integer	Ingress traffic (bit/s).
bps_attack	Long	Attack traffic (bit/s).
total_bps	Long	Total traffic.
pps_in	Long	Number of inbound packets per second.
pps_attack	Long	Number of attack packets per second.
total_pps	Long	Total inbound packet rate.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.

```
{
  "data" : [ {
    "period_start" : 1606188642720,
    "bps_in" : 0,
    "bps_attack" : 0,
    "total_bps" : 0,
    "pps_in" : 0,
    "pps_attack" : 0,
    "total_pps" : 0
  } ]
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class ListDailyReportSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
```



```
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ListDailyReportRequest request = new ListDailyReportRequest();
        request.withFloatingIpId("{floating_ip_id}");
        try {
            ListDailyReportResponse response = client.listDailyReport(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListDailyReportRequest()
        request.floating_ip_id = "{floating_ip_id}"
        response = client.list_daily_report(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
```



```
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListDailyReportRequest{}
    request.FloatingIpId = "{floating_ip_id}"
    response, err := client.ListDailyReport(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2.7 Querying Events of a Specified EIP

Function

This API allows you to query events of a specified EIP in the last 24 hours. Events include cleaning and black hole events. The query delay is no more than 5 minutes.

Calling Method

For details, see **Calling APIs**.

URI

GET /v1/{project_id}/antiddos/{floating_ip_id}/logs

Table 4-37 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.
floating_ip_id	Yes	String	ID corresponding to the EIP of a user.

Table 4-38 Query Parameters

Parameter	Mandatory	Type	Description
sort_dir	No	String	Values: • **desc**: descending order of time • **asc**: ascending order of time The default value is **desc**.
limit	No	String	Maximum number of returned results. The value ranges from 1 to 100. This parameter is used together with **offset**. If neither **limit** nor **offset** is specified, all servers are returned.
offset	No	String	Offset. This field is valid only if **limit** is specified.
ip	No	String	EIP of a user.

Request Parameters

Table 4-39 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-40 Response body parameters

Parameter	Type	Description
total	Long	Total number of EIPs.
logs	Array of **DailyLog** objects	Abnormal event list.

Table 4-41 DailyLog

Parameter	Type	Description
start_time	Long	Start time.
end_time	Long	End time.
status	Integer	Protection status. The options are as follows: • **1**: cleaning • **2**: blackhole
trigger_bps	Integer	Traffic at the triggering point.
trigger_pps	Integer	Packet rate at the triggering point.
trigger_http_pps	Integer	HTTP request rate at the triggering point.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.

```
{
  "total" : 0,
  "logs" : [ ]
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class ListDailyLogSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ListDailyLogRequest request = new ListDailyLogRequest();
        request.withFloatingIpId("{floating_ip_id}");
        try {
            ListDailyLogResponse response = client.listDailyLog(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
```



```
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListDailyLogRequest()
        request.floating_ip_id = "{floating_ip_id}"
        response = client.list_daily_log(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
```



```
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListDailyLogRequest{}
    request.FloatingIpId = "{floating_ip_id}"
    response, err := client.ListDailyLog(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2.8 Querying the Defense Status of a Specified EIP

Function

This API is used to query the defense status of a specified EIP.

Calling Method

For details, see **Calling APIs**.

URI

GET /v1/{project_id}/antiddos/{floating_ip_id}/status

Table 4-42 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Parameter	Mandatory	Type	Description
floating_ip_id	Yes	String	ID corresponding to the EIP of a user.

Table 4-43 Query Parameters

Parameter	Mandatory	Type	Description
ip	No	String	EIP of a user.

Request Parameters

Table 4-44 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-45 Response body parameters

Parameter	Type	Description
status	String	Protection status. The options are as follows: • **normal**: normal • **configging**: configuration in progress • **notConfig**: not configured • **packetcleaning**: cleaning • **packetdropping**: blackhole

Example Requests

None

Example Responses

Status code: 200

Request succeeded.

```
{
  "status" : "normal"
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class ShowDDosStatusSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowDDosStatusRequest request = new ShowDDosStatusRequest();
        request.withFloatingIpId("{floating_ip_id}");
        try {
            ShowDDosStatusResponse response = client.showDDosStatus(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
```



```
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowDDosStatusRequest()
        request.floating_ip_id = "{floating_ip_id}"
        response = client.show_d_dos_status(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
```



```
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowDDosStatusRequest{}
    request.FloatingIpId = "{floating_ip_id}"
    response, err := client.ShowDDosStatus(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2.9 Querying the List of a User's EIP Defense Statuses

Function

This API enables you to query the defense statuses of all EIPs, regardless whether an EIP has been bound to an Elastic Cloud Server (ECS) or not.

Calling Method

For details, see **Calling APIs**.

URI

GET /v2/{project_id}/antiddos

Table 4-46 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Table 4-47 Query Parameters

Parameter	Mandatory	Type	Description
status	No	String	Values: • **normal**: normal • **configging**: configuration in progress • **notConfig**: not configured • **packetcleaning**: cleaning • **packetdropping**: blackhole If this parameter is not specified, the defense statuses of all ECSs are displayed in the Neutron-queried order by default.
limit	No	String	Maximum number of returned results. The value ranges from 1 to 100.
offset	No	String	Offset. The value ranges from 0 to 2147483647.
ips	No	String	IP address, which can be in IPv4 or IPv6 format. Partial query is supported. For example, if you enter **?ip=192.168**, the EIP protection status of 192.168.111.1 and 10.192.168.8 will be returned.

Request Parameters

Table 4-48 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-49 Response body parameters

Parameter	Type	Description
total	Integer	Total number of EIPs.
ddosStatus	Array of **EipDDosStatus** objects	Protection status list.

Table 4-50 EipDDosStatus

Parameter	Type	Description
floating_ip_id	String	EIP ID.
floating_ip_address	String	Floating IP address.
product_type	String	EIP type. The value can be: • **Anti-DDoS**: The EIP is protected by Anti-DDoS. • **CNAD**: The EIP is protected by Cloud Native Anti-DDoS (CNAD).
status	String	Protection status. The options are as follows: • **normal**: normal • **configging**: configuration in progress • **notConfig**: not configured • **packetcleaning**: cleaning • **packetdropping**: blackhole
clean_threshold	Long	Cleaning threshold.
block_threshold	String	Blackhole threshold.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.


```
{
  "total" : 2,
  "ddosStatus" : [ {
    "floating_ip_id" : "xxxxxxxx-xxxx-4947-9fa8-dda843ae5d1d",
    "floating_ip_address" : "10.10.10.10",
    "status" : "normal",
    "product_type" : "CNAD",
    "clean_threshold" : 1000,
    "block_threshold" : ">=20Gbps"
  }, {
    "floating_ip_id" : "xxxxxxxx-xxxx-4d33-a2dc-2ace2d0283de",
    "floating_ip_address" : "11.11.11.11",
    "status" : "normal",
    "product_type" : "Anti-DDoS",
    "clean_threshold" : 120,
    "block_threshold" : ">=20Gbps"
  } ]
}
```


Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2.10 Enabling Anti-DDoS

Function

This API is used to enable Anti-DDoS.

Calling Method

For details, see **Calling APIs**.

URI

POST /v1/{project_id}/antiddos/{floating_ip_id}

Table 4-51 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.
floating_ip_id	Yes	String	ID corresponding to the EIP of a user.

Table 4-52 Query Parameters

Parameter	Mandatory	Type	Description
ip	No	String	EIP of a user.

Request Parameters

Table 4-53 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Table 4-54 Request body parameters

Parameter	Mandatory	Type	Description
app_type_id	Yes	Long	Application type ID. The value is fixed to **0**.
cleaning_access_pos_id	Yes	Long	Position ID of access limit during cleaning. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
enable_L7	Yes	Boolean	Whether to enable layer-7 protection. The value is fixed to **false**.
http_request_pos_id	Yes	Long	Segment ID of the HTTP request counts. The value is fixed to **1**.

Parameter	Mandatory	Type	Description
traffic_pos_id	Yes	Long	Position ID of traffic. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
antiddos_config_id	No	String	Protection level ID.

Response Parameters

Status code: 200

Table 4-55 Response body parameters

Parameter	Type	Description
error_code	String	Internal error code.
error_msg	String	Internal error description.
task_id	String	ID of a task. This ID can be used to query the status of the task. This field is reserved for future task audit extension and is not required currently.

Example Requests

Enable the Anti-DDoS policy for a specified EIP. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
POST https://{endpoint}/v1/{project_id}/antiddos/{floating_ip_id}

{
  "app_type_id" : 0,
  "cleaning_access_pos_id" : 8,
  "enable_L7" : false,
  "http_request_pos_id" : 1,
  "traffic_pos_id" : 1,
  "antiddos_config_name" : "test",
  "antiddos_config_id" : "1234567890"
}
```

Example Responses

Status code: 200

Request succeeded.

```
{
  "error_code" : "10000000",
  "error_msg" : "The task has been received and is being handled",
  "task_id" : "59385d2a-6266-4d3a-9122-a228c530f557"
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

Enable the Anti-DDoS policy for a specified EIP. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class EnableDefensePolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        EnableDefensePolicyRequest request = new EnableDefensePolicyRequest();
        request.withFloatingIpId("{floating_ip_id}");
        OpenAntiDDosServiceRequestBody body = new OpenAntiDDosServiceRequestBody();
        body.withTrafficPosId(1L);
        body.withHttpRequestPosId(1L);
        body.withEnableL7(false);
        body.withCleaningAccessPosId(8L);
        body.withAppTypeId(OpenAntiDDosServiceRequestBody.AppTypeIdEnum.NUMBER_0);
        request.withBody(body);
        try {
            EnableDefensePolicyResponse response = client.enableDefensePolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
```



```
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

Enable the Anti-DDoS policy for a specified EIP. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = EnableDefensePolicyRequest()
        request.floating_ip_id = "{floating_ip_id}"
        request.body = OpenAntiDDosServiceRequestBody(
            traffic_pos_id=1,
            http_request_pos_id=1,
            enable_l7=False,
            cleaning_access_pos_id=8,
            app_type_id=0
        )
        response = client.enable_defense_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

Enable the Anti-DDoS policy for a specified EIP. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
```



```
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.EnableDefensePolicyRequest{}
    request.FloatingIpId = "{floating_ip_id}"
    request.Body = &model.OpenAntiDDosServiceRequestBody{
        TrafficPosId: int64(1),
        HttpRequestPosId: int64(1),
        EnableL7: false,
        CleaningAccessPosId: int64(8),
        AppTypeId: model.GetOpenAntiDDosServiceRequestBodyAppTypeIdEnum().E_0,
    }
    response, err := client.EnableDefensePolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.2.11 Querying Quotas

Function

Query quotas.

Calling Method

For details, see **Calling APIs**.

URI

GET /v1/{project_id}/antiddos/quotas

Table 4-56 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Request Parameters

Table 4-57 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-58 Response body parameters

Parameter	Type	Description
ddos_quota	**ddos_quota** object	Quota information.

Table 4-59 ddos_quota

Parameter	Type	Description
current	Integer	Quota used by the current user.
quota	Integer	Maximum quota.

Example Requests

None

Example Responses

Status code: 200

OK

```
{
  "ddos_quota" : {
    "current" : 3,
    "quota" : 350
  }
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class ListQuotaSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
```



```
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ListQuotaRequest request = new ListQuotaRequest();
        try {
            ListQuotaResponse response = client.listQuota(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListQuotaRequest()
        response = client.list_quota(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)
```



```
func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListQuotaRequest{}
    response, err := client.ListQuota(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	OK
401	Unauthorized
403	Forbidden
404	Not Found

Error Codes

See **Error Codes**.

4.2.12 Updating All User Log Settings

Function

Updating all user log settings.

Calling Method

For details, see **Calling APIs**.

URI

PUT /v1/{project_id}/antiddos/lts-config

Table 4-60 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Table 4-61 Query Parameters

Parameter	Mandatory	Type	Description
enterprise_project_id	Yes	String	Enterprise project ID.

Request Parameters

Table 4-62 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Table 4-63 Request body parameters

Parameter	Mandatory	Type	Description
enabled	No	Boolean	Whether to enable logging.
lts_id_info	No	**lts_id_info** object	Log information.

Table 4-64 lts_id_info

Parameter	Mandatory	Type	Description
lts_group_id	No	String	Log group ID.
lts_attack_stream_id	No	String	Log stream ID.

Response Parameters

Status code: 200

OK

None

Example Requests

None

Example Responses

Status code: 200

OK

```
{
  "error_code" : "0",
  "error_msg" : "Success."
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class UpdateLogConfigSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
```



```
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateLogConfigRequest request = new UpdateLogConfigRequest();
        LtsConfigRequestAndResponse body = new LtsConfigRequestAndResponse();
        request.withBody(body);
        try {
            UpdateLogConfigResponse response = client.updateLogConfig(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = UpdateLogConfigRequest()
        request.body = LtsConfigRequestAndResponse(
        )
        response = client.update_log_config(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```


Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateLogConfigRequest{}
    request.Body = &model.LtsConfigRequestAndResponse{
    }
    response, err := client.UpdateLogConfig(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	OK
401	Unauthorized
403	Forbidden
404	Not Found

Error Codes

See **Error Codes**.

4.2.13 Querying All Log Settings

Function

Query all log settings.

Calling Method

For details, see **Calling APIs**.

URI

GET /v1/{project_id}/antiddos/lts-config

Table 4-65 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Table 4-66 Query Parameters

Parameter	Mandatory	Type	Description
enterprise_project_id	Yes	String	Enterprise project ID.

Request Parameters

Table 4-67 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-68 Response body parameters

Parameter	Type	Description
enabled	Boolean	Whether to enable logging.
lts_id_info	**lts_id_info** object	Log information.

Table 4-69 lts_id_info

Parameter	Type	Description
lts_group_id	String	Log group ID.
lts_attack_stream_id	String	Log stream ID.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.

```
{
  "enabled" : true,
  "lts_id_info" : {
    "lts_group_id" : "3f664a8c-53bd-4f77-922d-xxxxxxxxxxxx",
    "lts_attack_stream_id" : "ea9ffec3-6fe4-4884-9dd0-xxxxxxxxxxxx"
  }
}
```

Status Codes

Status Code	Description
200	Request succeeded.
401	Unauthorized
403	Forbidden
404	Not Found

Error Codes

See **Error Codes**.

4.3 Alarm Configuration Management

4.3.1 Querying Alarm Configuration

Function

You can query alarm configuration, such as whether a certain type of alarms will be received, and whether alarms are received through SMS messages or emails.

Calling Method

For details, see **Calling APIs**.

URI

GET /v2/{project_id}/warnalert/alertconfig/query

Table 4-70 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Request Parameters

Table 4-71 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-72 Response body parameters

Parameter	Type	Description
topic_urn	String	Unique ID of an alarm group.
display_name	String	Alarm group description.
warn_config	**warn_config** object	Alarm configuration information.

Table 4-73 warn_config

Parameter	Type	Description
antiDDoS	Boolean	DDoS attack.
back_doors	Boolean	Web shell.
bruce_force	Boolean	Brute-force attack (system logins, FTP, and DB).
high_privilege	Boolean	Excessive privileges assigned to a database process.
remote_login	Boolean	Alarms about remote logins.
send_frequency	Integer	**Range** • **0**: once a day • **1**: once every 30 minutes Mandatory for the HID.
waf	Boolean	Reserved field.
weak_password	Boolean	Weak password (system and database).

Example Requests

None

Example Responses

Status code: 200

Request succeeded.

```
{
  "warn_config" : {
    "antiDDoS" : false,
    "bruce_force" : false,
    "remote_login" : false,
    "weak_password" : false,
    "high_privilege" : false,
    "back_doors" : false,
```



```
    "waf" : false,
    "send_frequency" : 0
  },
  "topic_urn" : null,
  "display_name" : null
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v2.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v2.*;
import com.huaweicloud.sdk.antiddos.v2.model.*;


public class ShowAlertConfigSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowAlertConfigRequest request = new ShowAlertConfigRequest();
        try {
            ShowAlertConfigResponse response = client.showAlertConfig(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8
```



```
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v2.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowAlertConfigRequest()
        response = client.show_alert_config(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowAlertConfigRequest{}
    response, err := client.ShowAlertConfig(request)
```



```
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.3.2 Updating Alarm Configuration

Function

This API allows you to update alarm configuration, such as whether a certain type of alarms will be received, and whether alarms are received through SMS messages or emails.

Calling Method

For details, see **Calling APIs**.

URI

POST /v2/{project_id}/warnalert/alertconfig/update

Table 4-74 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Request Parameters

Table 4-75 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Table 4-76 Request body parameters

Parameter	Mandatory	Type	Description
display_name	Yes	String	Alarm group description.
topic_urn	Yes	String	Unique ID of an alarm group.
warn_config	Yes	**WarnConfig** object	Alarm configuration.

Table 4-77 WarnConfig

Parameter	Mandatory	Type	Description
antiDDoS	Yes	Boolean	DDoS attack.
back_doors	No	Boolean	Web shell.
bruce_force	No	Boolean	Brute-force attack (system logins, FTP, and DB).
high_privilege	No	Boolean	Excessive privileges assigned to a database process.
remote_login	No	Boolean	Alarms about remote logins.
send_frequency	No	Integer	**Range** • **0**: once a day • **1**: once every 30 minutes Mandatory for the HID.
waf	No	Boolean	Reserved field.
weak_password	No	Boolean	Weak password (system and database).

Response Parameters

Status code: 200

Table 4-78 Response body parameters

Parameter	Type	Description
error_code	String	Internal error code.
error_msg	String	Internal error description.

Example Requests

Set the SMN topic to **urn:smn:cn-north-7:2d2d90a56a3243bdb909f6a24a27be8d:cnad-test-intl** and set the attack notification type to DDoS attacks.

```
POST https://{endpoint}/v2/{project_id}/warnalert/alertconfig/update

{
  "display_name" : "",
  "topic_urn" : "urn:smn:cn-north-7:2d2d90a56a3243bdb909f6a24a27be8d:cnad-test-intl",
  "warn_config" : {
    "antiDDoS" : true,
    "back_doors" : false,
    "bruce_force" : false,
    "high_privilege" : false,
    "remote_login" : false,
    "send_frequency" : 1,
    "waf" : false,
    "weak_password" : false
  }
}
```

Example Responses

Status code: 200

Request succeeded.

```
{
  "error_code" : "10000000",
  "error_msg" : "Ok",
  "task_id" : ""
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

Set the SMN topic to **urn:smn:cn-north-7:2d2d90a56a3243bdb909f6a24a27be8d:cnad-test-intl** and set the attack notification type to DDoS attacks.

```
package com.huaweicloud.sdk.test;
```



```
import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v2.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v2.*;
import com.huaweicloud.sdk.antiddos.v2.model.*;


public class UpdateAlertConfigSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateAlertConfigRequest request = new UpdateAlertConfigRequest();
        UpdateAlertConfigRequestBody body = new UpdateAlertConfigRequestBody();
        WarnConfig warnConfigbody = new WarnConfig();
        warnConfigbody.withAntiDDoS(true)
            .withBackDoors(false)
            .withBruceForce(false)
            .withHighPrivilege(false)
            .withRemoteLogin(false)
            .withSendFrequency(WarnConfig.SendFrequencyEnum.NUMBER_1)
            .withWaf(false)
            .withWeakPassword(false);
        body.withWarnConfig(warnConfigbody);
        body.withTopicUrn("urn:smn:cn-north-7:2d2d90a56a3243bdb909f6a24a27be8d:cnad-test-intl");
        body.withDisplayName("");
        request.withBody(body);
        try {
            UpdateAlertConfigResponse response = client.updateAlertConfig(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

Set the SMN topic to **urn:smn:cn-north-7:2d2d90a56a3243bdb909f6a24a27be8d:cnad-test-intl** and set the attack notification type to DDoS attacks.


```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v2.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = UpdateAlertConfigRequest()
        warnConfigbody = WarnConfig(
            anti_d_do_s=True,
            back_doors=False,
            bruce_force=False,
            high_privilege=False,
            remote_login=False,
            send_frequency=1,
            waf=False,
            weak_password=False
        )
        request.body = UpdateAlertConfigRequestBody(
            warn_config=warnConfigbody,
            topic_urn="urn:smn:cn-north-7:2d2d90a56a3243bdb909f6a24a27be8d:cnad-test-intl",
            display_name=""
        )
        response = client.update_alert_config(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

Set the SMN topic to **urn:smn:cn-north-7:2d2d90a56a3243bdb909f6a24a27be8d:cnad-test-intl** and set the attack notification type to DDoS attacks.

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
```



```
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateAlertConfigRequest{}
    backDoorsWarnConfig:= false
    bruceForceWarnConfig:= false
    highPrivilegeWarnConfig:= false
    remoteLoginWarnConfig:= false
    sendFrequencyWarnConfig:= model.GetWarnConfigSendFrequencyEnum().E_1
    wafWarnConfig:= false
    weakPasswordWarnConfig:= false
    warnConfigbody := &model.WarnConfig{
        AntiDDoS: true,
        BackDoors: &backDoorsWarnConfig,
        BruceForce: &bruceForceWarnConfig,
        HighPrivilege: &highPrivilegeWarnConfig,
        RemoteLogin: &remoteLoginWarnConfig,
        SendFrequency: &sendFrequencyWarnConfig,
        Waf: &wafWarnConfig,
        WeakPassword: &weakPasswordWarnConfig,
    }
    request.Body = &model.UpdateAlertConfigRequestBody{
        WarnConfig: warnConfigbody,
        TopicUrn: "urn:smn:cn-north-7:2d2d90a56a3243bdb909f6a24a27be8d:cnad-test-intl",
        DisplayName: "",
    }
    response, err := client.UpdateAlertConfig(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.4 Default Protection Policy Management

4.4.1 Configuring the Default Anti-DDoS Protection Policy

Function

This API is used to configure the default protection policy. After a protection policy is configured, it applies to the newly purchased EIPs.

Calling Method

For details, see **Calling APIs**.

URI

POST /v1/{project_id}/antiddos/default-config

Table 4-79 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Request Parameters

Table 4-80 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Table 4-81 Request body parameters

Parameter	Mandatory	Type	Description
enable_L7	Yes	Boolean	Whether to enable layer-7 protection. The value is fixed to **false**.
traffic_pos_id	Yes	Long	Position ID of traffic. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
http_request_pos_id	Yes	Long	Segment ID of the HTTP request counts. The value is fixed to **1**.
cleaning_access_pos_id	Yes	Long	Position ID of access limit during cleaning. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
app_type_id	Yes	Long	Application type ID. The value is fixed to **0**.
antiddos_config_id	No	String	Protection level ID.

Response Parameters

Status code: 200

Table 4-82 Response body parameters

Parameter	Type	Description
error_code	String	Internal error code.
error_msg	String	Internal error description.

Example Requests

Configure the default protection policy a the user. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
POST https://{endpoint}/v1/{project_id}/antiddos/default-config

{
  "app_type_id" : 0,
  "cleaning_access_pos_id" : 8,
  "enable_L7" : false,
  "http_request_pos_id" : 1,
  "traffic_pos_id" : 1,
  "antiddos_config_name" : "test",
  "antiddos_config_id" : "1234567890"
}
```

Example Responses

Status code: 200

Request succeeded.

```
{
  "error_code" : "10000000",
  "error_msg" : "Ok",
  "task_id" : ""
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

Configure the default protection policy a the user. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class CreateDefaultConfigSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
```



```
            .withSk(sk);

    AntiDDoSClient client = AntiDDoSClient.newBuilder()
            .withCredential(auth)
            .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
            .build();
    CreateDefaultConfigRequest request = new CreateDefaultConfigRequest();
    DdosConfig body = new DdosConfig();
    body.withAppTypeId(DdosConfig.AppTypeIdEnum.NUMBER_0);
    body.withCleaningAccessPosId(8L);
    body.withHttpRequestPosId(1L);
    body.withTrafficPosId(1L);
    body.withEnableL7(false);
    request.withBody(body);
    try {
        CreateDefaultConfigResponse response = client.createDefaultConfig(request);
        System.out.println(response.toString());
    } catch (ConnectionException e) {
        e.printStackTrace();
    } catch (RequestTimeoutException e) {
        e.printStackTrace();
    } catch (ServiceResponseException e) {
        e.printStackTrace();
        System.out.println(e.getHttpStatusCode());
        System.out.println(e.getRequestId());
        System.out.println(e.getErrorCode());
        System.out.println(e.getErrorMsg());
    }
  }
}
```

Python

Configure the default protection policy a the user. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateDefaultConfigRequest()
        request.body = DdosConfig(
            app_type_id=0,
            cleaning_access_pos_id=8,
            http_request_pos_id=1,
            traffic_pos_id=1,
            enable_l7=False
```



```
        )
        response = client.create_default_config(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

Configure the default protection policy a the user. Set the position ID of access limit during cleaning to 8, and set the position ID of traffic to 1.

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateDefaultConfigRequest{}
    request.Body = &model.DdosConfig{
        AppTypeId: model.GetDdosConfigAppTypeIdEnum().E_0,
        CleaningAccessPosId: int64(8),
        HttpRequestPosId: int64(1),
        TrafficPosId: int64(1),
        EnableL7: false,
    }
    response, err := client.CreateDefaultConfig(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.4.2 Querying the Default Ani-DDoS Protection Policy

Function

This API is used to query the default protection policy configured by a user.

Calling Method

For details, see **Calling APIs**.

URI

GET /v1/{project_id}/antiddos/default-config

Table 4-83 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Request Parameters

Table 4-84 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-85 Response body parameters

Parameter	Type	Description
enable_L7	Boolean	Whether to enable layer-7 protection. The value is fixed to **false**.
traffic_pos_id	Long	Position ID of traffic. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
http_request_pos_id	Long	Segment ID of the HTTP request counts. The value is fixed to **1**.
cleaning_access_pos_id	Long	Position ID of access limit during cleaning. The options are as follows: **1**: 10 Mbit/s; **2**: 30 Mbit/s; **3**: 50 Mbit/s; **4**: 70 Mbit/s; **5**: 100 Mbit/s; **6**: 150 Mbit/s; **7**: 200 Mbit/s; **8**: 250 Mbit/s; **9**: 300 Mbit/s; **10**: 500 Mbit/s; **11**: 800 Mbit/s;**88**: 1,000 Mbit/s; **99**: default protection
app_type_id	Long	Application type ID. The value is fixed to **0**.
antiddos_config_name	String	Protection level name.
antiddos_config_id	String	Protection level ID.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.

```
{
  "app_type_id" : 1,
  "cleaning_access_pos_id" : 8,
  "enable_L7" : false,
  "http_request_pos_id" : 8,
  "traffic_pos_id" : 8,
```



```
  "antiddos_config_name" : "test",
  "antiddos_config_id" : "1234567890"
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class ShowDefaultConfigSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowDefaultConfigRequest request = new ShowDefaultConfigRequest();
        try {
            ShowDefaultConfigResponse response = client.showDefaultConfig(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
```



```
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowDefaultConfigRequest()
        response = client.show_default_config(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowDefaultConfigRequest{}
    response, err := client.ShowDefaultConfig(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
```



```
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

4.4.3 Deleting the default Anti-DDoS Policies

Function

This API is used to delete the default protection policy configured by a user.

Calling Method

For details, see **Calling APIs**.

URI

DELETE /v1/{project_id}/antiddos/default-config

Table 4-86 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID.

Request Parameters

Table 4-87 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. It can be obtained by calling the IAM API used to obtain a user token. The value of **X-Subject-Token** in the response header is the user token.
Content-Type	Yes	String	Content-Type request header.

Response Parameters

Status code: 200

Table 4-88 Response body parameters

Parameter	Type	Description
error_code	String	Internal error code.
error_msg	String	Internal error description.

Example Requests

None

Example Responses

Status code: 200

Request succeeded.

```
{
  "error_code" : "10000000",
  "error_msg" : "Ok",
  "task_id" : ""
}
```

SDK Sample Code

The SDK sample code is as follows.

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
```



```
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.antiddos.v1.region.AntiDDoSRegion;
import com.huaweicloud.sdk.antiddos.v1.*;
import com.huaweicloud.sdk.antiddos.v1.model.*;


public class DeleteDefaultConfigSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        AntiDDoSClient client = AntiDDoSClient.newBuilder()
                .withCredential(auth)
                .withRegion(AntiDDoSRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteDefaultConfigRequest request = new DeleteDefaultConfigRequest();
        try {
            DeleteDefaultConfigResponse response = client.deleteDefaultConfig(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkantiddos.v1.region.antiddos_region import AntiDDoSRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkantiddos.v1 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)
```



```
    client = AntiDDoSClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(AntiDDoSRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = DeleteDefaultConfigRequest()
        response = client.delete_default_config(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    antiddos "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/antiddos/v1/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := antiddos.NewAntiDDoSClient(
        antiddos.AntiDDoSClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.DeleteDefaultConfigRequest{}
    response, err := client.DeleteDefaultConfig(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

More

For SDK sample code of more programming languages, see the Sample Code tab in **API Explorer**. SDK sample code can be automatically generated.

Status Codes

Status Code	Description
200	Request succeeded.

Error Codes

See **Error Codes**.

5 Anti-DDoS Permissions and Supported Actions

5.1 Permissions and Supported Actions

You can use Identity and Access Management (IAM) for fine-grained permissions management of your Anti-DDoS resources. If your Huawei Cloud account does not need individual IAM users, you can skip this section.

With IAM, you can control access to specific Huawei Cloud resources from principals (IAM users, user groups, agencies, or trust agencies). IAM supports role/policy-based authorization and identity policy-based authorization.

The following table describes the main differences.

Table 5-1 Differences between role/policy-based authorization and identity policy-based authorization

Name	Core Relationship	Permission	Authorization Method	Scenario
Role/Policy-based authorization	User-permission-authorization scope	• System-defined roles • System-defined policies • Custom policies	Assigning roles or policies to principals	To authorize a user, you need to add it to a user group first and then specify the scope of authorization. It provides a limited number of condition keys and cannot meet the requirements of fine-grained permissions control. This method is suitable for small- and medium-sized enterprises.

Name	Core Relationship	Permission	Authorization Method	Scenario
Identity policy-based authorization	User-policy	• System-defined identity policies • Custom identity policies	• Assigning identity policies to principals • Attaching identity policies to principals	You can authorize a user by directly attaching an identity policy to it. You can customize policies and attach them to specified users. Identity policies allow you to perform refined access control more efficiently and flexibly. However, this model is more complex and requires higher personnel expertise. It is more suitable for medium- and large-sized enterprises.

Assume that you want to grant IAM users the permission to create ECSs in CN North-Beijing4 and OBS buckets in CN South-Guangzhou. With role/policy-based authorization, the administrator needs to create two custom policies and attach both to the IAM users. With identity policy-based authorization, the administrator only needs to create one custom identity policy and configure the condition key **g:RequestedRegion** for the policy, and then attach the policy to the principals or grant the principals the access permissions to the specified regions. Identity policy-based authorization is more flexible than role/policy-based authorization.

Policies/identity policies and actions in the two authorization models are not interoperable. You are advised to use the identity policy-based authorization model.

If you use IAM users to call an API, the IAM users must be granted the required permissions. The required permissions are determined by the actions supported by the API. Only users with the policies allowing for those actions can call the API successfully.

5.2 Actions Supported by Policy-based Authorization

This section describes the actions supported policy-based authorization for Anti-DDoS.

Supported Actions

Anti-DDoS provides system-defined policies that can be directly used in IAM. You can also create custom policies and use them to supplement system-defined policies, implementing more refined access control. Operations supported by policies are specific to APIs. The following are common concepts related to policies:

- Permissions: statements in a policy that allow or deny certain operations APIs: REST APIs that can be called by a user who has been granted specific permissions.
- Actions: Specific operations that are allowed or denied in a custom policy.
- Dependencies: Actions which a specific action depends on. When allowing an action for a user, you also need to allow its dependent actions for that user.
- IAM or enterprise projects: Type of projects for which an action will take effect. For example, if you set the authorization scope of a custom policy to both IAM projects and enterprise projects, the policy takes effect for user groups in either IAM or enterprise projects. Policies that contain actions only for IAM projects can be used and applied to IAM only. Administrators can check whether an action supports IAM projects or enterprise projects in the action list. "√" indicates that the action supports the project and "×" indicates that the action does not support the project. For details about the differences between IAM and enterprise management, see **What Are the Differences Between IAM and Enterprise Management?**

Table 5-2 lists the API actions supported by Anti-DDoS.

Table 5-2 Supported actions

Permission	API	Action	IAM Project	Enterprise Project
Querying Anti-DDoS tasks	GET /v2/{project_id}/query-task-status	anti-ddos:task:list	√	√
Querying the list of defense statuses of EIPs	GET /v1/{project_id}/antiddos	anti-ddos:ip:listDefenseStatuses	√	√
Querying Anti-DDoS specification range	GET /v2/{project_id}/antiddos/query-config-list	anti-ddos:optionalDefensePolicy:list	√	√
Querying weekly defense statistics	GET /v1/{project_id}/antiddos/weekly	anti-ddos:ip:getWeeklyReport	√	√
Querying configured Anti-DDoS policies	GET /v1/{project_id}/antiddos/{floating_ip_id}	anti-ddos:ip:getDefensePolicy	√	√
Updating Anti-DDoS policies	PUT /v1/{project_id}/antiddos/{floating_ip_id}	anti-ddos:ip:updateDefensePolicy	√	√

Permission	API	Action	IAM Project	Enterprise Project
Querying the traffic of a specified EIP	GET /v1/{project_id}/antiddos/{floating_ip_id}/daily	anti-ddos:ip:getDailyTrafficReport	√	√
Querying events of a specified EIP	GET /v1/{project_id}/antiddos/{floating_ip_id}/logs	anti-ddos:ip:getDailyEventReport	√	√
Querying the defense status of a specified EIP	GET /v1/{project_id}/antiddos/{floating_ip_id}/status	anti-ddos:ip:getDefenseStatus	√	√
Querying the list of defense statuses of EIPs	GET /v2/{project_id}/antiddos	anti-ddos:ip:listDefenseStatuses	√	√
Enabling Anti-DDoS	POST /v1/{project_id}/antiddos/{floating_ip_id}	anti-ddos:ip:enableDefensePolicy	√	√
Querying the user quota	GET /v1/{project_id}/antiddos/quotas	anti-ddos:quota:list	√	√
Updating all user log settings	PUT /v1/{project_id}/antiddos/lts-config	anti-ddos:logConfig:update	√	√
Querying all log settings	GET /v1/{project_id}/antiddos/lts-config	anti-ddos:logConfig:get	√	√
Querying the alarm configuration	GET /v2/{project_id}/warnalert/alertconfig/query	anti-ddos:alertConfig:get	√	√
Updating the alarm configuration	POST /v2/{project_id}/warnalert/alertconfig/update	anti-ddos:alertConfig:update	√	√
Configuring the default Anti-DDoS protection policy	POST /v1/{project_id}/antiddos/default-config	anti-ddos:defaultDefensePolicy:create	√	√

Permission	API	Action	IAM Project	Enterprise Project
Querying the default Anti-DDoS protection policy	GET /v1/{project_id}/antiddos/default-config	anti-ddos:defaultDefensePolicy:get	√	√
Deleting the default Anti-DDoS policy	DELETE /v1/{project_id}/antiddos/default-config	anti-ddos:defaultDefensePolicy:delete	√	√

5.3 Actions Supported by Identity Policy-based Authorization

IAM provides system-defined identity policies to define typical cloud service permissions. You can also create custom identity policies using the actions supported by cloud services for more refined access control.

In addition to IAM, the **Organizations** service also provides **Service Control Policies (SCPs)** to set access control policies.

SCPs do not actually grant any permissions to an entity. They only set the permissions boundary for the entity. When SCPs are attached to an organizational unit (OU) or a member account, the SCPs do not directly grant permissions to that OU or member account. Instead, the SCPs only determine what permissions are available for that member account or those member accounts under that OU. The granted permissions can be applied only if they are allowed by the SCPs.

To learn more about how IAM is different from Organizations for access control, see **How IAM Is Different from Organizations for Access Control?**.

This section describes the elements used by IAM custom identity policies and Organizations SCPs. The elements include actions, resources, and conditions.

- For details about how to use these elements to edit an IAM custom identity policy, see **Creating a Custom Identity Policy**.
- For details about how to use these elements to edit a custom SCP, see **Creating an SCP**.

Actions

Actions are specific operations that are allowed or denied in an identity policy.

- The Access Level column describes how the action is classified (List, Read, or Write). This classification helps you understand the level of access that an action grants when you use it in an identity policy.
- The Resource Type column indicates whether the action supports resource-level permissions.

  - You can use a wildcard (*) to indicate all resource types. If this column is empty (-), the action does not support resource-level permissions and you must specify all resources ("*") in your identity policy statements.
  - If this column includes a resource type, you must specify the URN in the Resource element of your identity policy statements.
  - Required resources are marked with asterisks (*) in the table. If you specify a resource in a statement using this action, then it must be of this type.

  For details about the resource types defined by Anti-DDoS, see **Resources**.
- The Condition Key column contains keys that you can specify in the Condition element of an identity policy statement.
  - If the Resource Type column has values for an action, the condition key takes effect only for the listed resource types.
  - If the Resource Type column is empty (-) for an action, the condition key takes effect for all resources that action supports.
  - If the Condition Key column is empty (-) for an action, the action does not support any condition keys.

  For details about the condition keys defined by Anti-DDoS, see **Conditions**.
- The Alias column lists the policy actions that are configured in identity policies. With these actions, you can use APIs for policy-based authorization. For details, see **Policies and Identity Policies**.

The following table lists the actions that you can define in identity policy statements for Anti-DDoS.

Table 5-3 Actions supported by Anti-DDoS

Action	Description	Access Level	Resource Type (*: required)	Condition Key	Alias
anti-ddos:task:list	Grants permission to query Anti-DDoS tasks.	List	-	-	-
anti-ddos:quota:list	Grants permission to query quotas.	List	-	-	-
anti-ddos:optionalDefensePolicy:list	Grants permission to query Anti-DDoS protection specifications.	List	-	-	-

Action	Description	Access Level	Resource Type (*: required)	Condition Key	Alias
anti-ddos:logConfig:update	Grants permission to update LTS configurations.	Write	-	-	-
anti-ddos:logConfig:get	Grants permission to query LTS configurations.	Read	-	-	-
anti-ddos:ip:updateDefensePolicy	Grants permission to update Anti-DDoS.	Write	ip *	• g:ResourceTag/<tag-key> • g:EnterpriseProjectId	-
anti-ddos:ip:untagResource	Grants permission to delete tags in batches.	Write	ip *	• g:ResourceTag/<tag-key> • g:EnterpriseProjectId	-
anti-ddos:ip:tagResource	Grants permission to add tags in batches.	Write	ip *	• g:ResourceTag/<tag-key> • g:EnterpriseProjectId	-
anti-ddos:ip:listTagsForResource	Grants permission to query resource tags.	List	ip *	-	-
anti-ddos:ip:listDefenseStatuses	Grants permission to query EIP protection statuses.	List	ip *	-	-

Action	Description	Access Level	Resource Type (*: required)	Condition Key	Alias
anti-ddos:ip:getWeeklyReport	Grants permission to query weekly protection statistics.	Read	ip *	• g:ResourceTag/<tag-key> • g:EnterpriseProjectId	-
anti-ddos:ip:getDefenseStatus	Grants permission to query the protection status of an EIP.	Read	ip *	• g:ResourceTag/<tag-key> • g:EnterpriseProjectId	-
anti-ddos:ip:getDefensePolicy	Grants permission to query Anti-DDoS.	Read	ip *	• g:ResourceTag/<tag-key> • g:EnterpriseProjectId	-
anti-ddos:ip:getDailyTrafficReport	Grants permission to query protected traffic of an EIP.	Read	ip *	• g:ResourceTag/<tag-key> • g:EnterpriseProjectId	-
anti-ddos:ip:getDailyEventReport	Grants permission to query abnormal events of an EIP.	Read	ip *	• g:ResourceTag/<tag-key> • g:EnterpriseProjectId	-
anti-ddos:ip:enableDefensePolicy	Grants permission to enable Anti-DDoS.	Write	ip *	• g:ResourceTag/<tag-key> • g:EnterpriseProjectId	-

Action	Description	Access Level	Resource Type (*: required)	Condition Key	Alias
anti-ddos:ip:disableDefensePolicy	Grants permission to disable Anti-DDoS.	Write	ip *	• g:ResourceTag/<tag-key> • g:EnterpriseProjectId	-
anti-ddos:defaultDefensePolicy:get	Grants permission to query default Anti-DDoS protection policies.	Read	-	-	-
anti-ddos:defaultDefensePolicy:delete	Grants permission to delete default Anti-DDoS protection policies.	Write	-	-	-
anti-ddos:defaultDefensePolicy:create	Grants permission to configure default Anti-DDoS protection policies.	Write	-	-	-
anti-ddos:alertConfig:update	Grants permission to update alarm configurations.	Write	-	-	-
anti-ddos:alertConfig:get	Grants permission to query alarm configurations.	Read	-	-	-

Each API of Anti-DDoS usually supports one or more actions. **Table 5-4** lists the supported actions and dependencies.

Table 5-4 Actions and dependencies supported by Anti-DDoS APIs

API	Action	Dependencies
GET /v2/{project_id}/query-task-status	anti-ddos:task:list	-
GET /v1/{project_id}/antiddos/quotas	anti-ddos:quota:list	-
GET /v1/{project_id}/antiddos/query-config-list	anti-ddos:optionalDefense-Policy:list	-
PUT /v1/{project_id}/antiddos/lts-config	anti-ddos:logConfig:update	-
GET /v1/{project_id}/antiddos/lts-config	anti-ddos:logConfig:get	-
PUT /v1/{project_id}/antiddos/{floating_ip_id}	anti-ddos:ip:updateDefensePoli-cy	-
DELETE /v1/{project_id}/antiddos-ip/{resource_id}/tags/delete	anti-ddos:ip:untagResource	-
POST /v1/{project_id}/antiddos-ip/{resource_id}/tags/create	anti-ddos:ip:tagResource	-
GET /v1/{project_id}/antiddos-ip/{resource_id}/tags	anti-ddos:ip:listTagsForResource	-
GET /v1/{project_id}/antiddos-ip/tags	anti-ddos:ip:listTagsForResource	-
GET /v1/{project_id}/antiddos	anti-ddos:ip:listDefenseStatuses	-

API	Action	Dependencies
POST /v1/{project_id}/antiddos-ip/resource-instances/count	anti-ddos:ip:listDefenseStatuses	-
POST /v1/{project_id}/antiddos-ip/resource-instances/filter	anti-ddos:ip:listDefenseStatuses	-
GET /v1/{project_id}/antiddos/weekly	anti-ddos:ip:getWeeklyReport	-
GET /v1/{project_id}/antiddos/weekly-export	anti-ddos:ip:getWeeklyReport	-
GET /v1/{project_id}/antiddos/{floating_ip_id}/status	anti-ddos:ip:getDefenseStatus	-
GET /v1/{project_id}/antiddos/{floating_ip_id}	anti-ddos:ip:getDefensePolicy	-
GET /v1/{project_id}/antiddos/queryIsEnabledResult/query	anti-ddos:ip:getDefensePolicy	-
GET /v1/{project_id}/antiddos/{floating_ip_id}/daily	anti-ddos:ip:getDailyTrafficReport	-
GET /v1/{project_id}/antiddos/{floating_ip_id}/daily-export	anti-ddos:ip:getDailyTrafficReport	-

API	Action	Dependencies
GET /v1/{project_id}/antiddos/{floating_ip_id}/logs	anti-ddos:ip:getDailyEventReport	-
POST /v1/{project_id}/antiddos/{floating_ip_id}	anti-ddos:ip:enableDefensePolicy	-
POST /v1/{project_id}/antiddos/immediate_protection	anti-ddos:ip:enableDefensePolicy	-
DELETE /v1/{project_id}/antiddos/{floating_ip_id}	anti-ddos:ip:disableDefensePolicy	-
DELETE /v1/{project_id}/antiddos/{floating_ip_id}/closeAndReason	anti-ddos:ip:disableDefensePolicy	-
GET /v1/{project_id}/antiddos/default/config	anti-ddos:defaultDefensePolicy:get	-
DELETE /v1/{project_id}/antiddos/default/config	anti-ddos:defaultDefensePolicy:delete	-
POST /v1/{project_id}/antiddos/default/config	anti-ddos:defaultDefensePolicy:create	-
POST /v2/{project_id}/warnalert/alertconfig/update	anti-ddos:alertConfig:update	-
GET /v2/{project_id}/warnalert/alertconfig/query	anti-ddos:alertConfig:get	-

Resources

A resource type indicates the resources that an identity policy applies to. If you specify a resource type for any action in **Table 5-5**, the resource URN must be specified in the identity policy statements using that action, and the identity policy applies only to resources of this type. If no resource type is specified, the Resource element is marked with an asterisk (*) and the identity policy applies to all resources. You can also set condition keys in an identity policy to define resource types.

The following table lists the resource types that you can define in identity policy statements for Anti-DDoS.

Table 5-5 Resource types supported by Anti-DDoS

Resource Type	URN
ip	anti-ddos:<region>:<account-id>:ip:<ip-id>

Conditions

Anti-DDoS does not support service-specific condition keys in identity policies.It can only use global condition keys applicable to all services. For details, see **Global Condition Keys**.

A Appendix

A.1 HTTP Status Codes of Cloud Native Anti-DDoS Basic

- Normal

Returned Value	Description
200	The request is successfully processed.

- Abnormal

Status Code	Status	Description
400	Bad Request	The server fails to process the request.
401	Unauthorized	The requested page requires a username and a password.
403	Forbidden	Access to the requested page is denied.
404	Not Found	The server fails to find the requested page.
405	Method Not Allowed	Method specified in the request is not allowed.
406	Not Acceptable	Response generated by the server is not acceptable to the client.
407	Proxy Authentication Required	Proxy authentication is required before the request is processed.
408	Request Timeout	A timeout error occurs because the request is not processed within the specified waiting period of the server.

Status Code	Status	Description
409	Conflict	The request cannot be processed due to a conflict.
500	Internal Server Error	The request is not processed due to a server error.
501	Not Implemented	The request is not processed because the server does not support the requested function.
502	Bad Gateway	The request is not processed, and the server receives an invalid response from the upstream server.
503	Service Unavailable	The request is not processed due to a temporary system abnormality.
504	Gateway Timeout	A gateway timeout error occurs.

A.2 Obtaining a Project ID

Obtaining a Project ID by Calling an API

You can obtain the project ID by calling the API for **Querying Project Information Based on Specified Criteria**.

The API used to obtain a project ID is GET https://{Endpoint}/v3/projects. **{Endpoint}** is the IAM endpoint and can be obtained from **Regions and Endpoints**. For details about API authentication, see **Authentication**.

In the following example, **id** indicates the project ID.

```
{
    "projects": [
        {
            "domain_id": "65382450e8f64ac0870cd180d14e684b",
            "is_domain": false,
            "parent_id": "65382450e8f64ac0870cd180d14e684b",
            "name": "xxxxxxxx",
            "description": "",
            "links": {
                "next": null,
                "previous": null,
                "self": "https://www.example.com/v3/projects/a4a5d4098fb4474fa22cd05f897d6b99"
            },
            "id": "a4a5d4098fb4474fa22cd05f897d6b99",
            "enabled": true
        }
    ],
    "links": {
        "next": null,
        "previous": null,
        "self": "https://www.example.com/v3/projects"
    }
}
```


Obtaining a Project ID from the Console

A project ID is required for some URLs when an API is called. To obtain a project ID, perform the following operations:

1. Log in to the management console.
2. Click the username and choose **My Credentials** from the drop-down list.
3. On the **API Credentials** page, view the project ID in the project list.

Figure A-1 Viewing project IDs