Anti-DDoS

API Reference

Issue 01

Date 2024-09-12

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 API Usage Guidelines

Public cloud APIs comply with the RESTful API design principles. REST-based Web services are organized into resources. Each resource is identified by one or more Uniform Resource Identifiers (URIs). An application accesses a resource based on the resource's Unified Resource Locator (URL). A URL is usually in the following format: *https://Endpoint/uri*. In the URL, **uri** indicates the resource path, that is, the API access path.

Public cloud APIs use HTTPS as the transmission protocol. Requests/Responses are transmitted by using JSON messages, with media type represented by **Application/json**.

2 Before You Start

2.1 Overview

The Anti-DDoS service protects public IP addresses against Layer 4 to Layer 7 distributed denial of service (DDoS) attacks and sends alarms immediately when detecting an attack. Anti-DDoS improves the bandwidth utilization and ensures the stable running of user services.

This document describes how to use application programming interfaces (APIs) to perform operations to Anti-DDoS, such as querying or updating Anti-DDoS protection policy. For details about all supported operations, see **API Overview**.

Before calling Anti-DDoS APIs, ensure that you are familiar with Anti-DDoS concepts. For details, see section "Service Overview" of the *Anti-DDoS User Guide*.

2.2 API Calling

Anti-DDoS provides Representational State Transfer (REST) APIs, allowing you to use HTTPS requests to call them. For details, see **API Calling**.

2.3 Notes and Constraints

For details about the constraints, see the API description.

3 API Overview

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

Type	Description
APIs of Anti-DDoS	Anti-DDoS APIs, including the APIs for querying and updating the Anti-DDoS defense policies.
Alarm Reminding APIs	Alarm reminding APIs, including the API for querying alarm configuration information

4 API Calling

4.1 Making an API Request

This section describes the structure of a REST API request, and uses the IAM API for obtaining a user token as an example to demonstrate how to call an API. The obtained token can then be used to authenticate the calling of other APIs.

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 the administrator.
- **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 used to obtain a user token is **/v3/auth/tokens**.
- **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.

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 case of the API used to obtain a user token, the request method is POST. The request is as follows:

```
POST https://{{endpoint}}/v3/auth/tokens
```

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.
- **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.

  NOTE

  In addition to supporting token-based authentication, APIs also support authentication using access key ID/secret access key (AK/SK). During AK/SK-based authentication, an SDK is used to sign the request, and the **Authorization** (signature information) and **X-Sdk-Date** (time when the request is sent) header fields are automatically added to the request.

  For more information, see **AK/SK-based Authentication**.

The API used to obtain a user token does not require authentication. Therefore, only the **Content-Type** field needs to be added to requests for calling the API. An example of such requests is as follows:

Request Body

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

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.

In the case of the API used to obtain a user token, the request parameters and parameter description can be obtained from the API request. The following provides an example request with a body included. Set ***username*** to the name of a user, ***domainname*** to the name of the account that the user belongs to, ******** to the user's login password, and ***xxxxxxxxxxxxxxxxx*** to the project name. You can learn more information about projects from .

NOTE

The **scope** parameter specifies where a token takes effect. You can set **scope** to an account or a project under an account. In the following example, the token takes effect only for the resources in a specified project. For more information about this API, see "Obtaining a User Token".

```
POST https://{{endpoint}}/v3/auth/tokens
Content-Type: application/json
{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "username",
                    "password": "********",
                    "domain": {
                        "name": "domainname"
                    }
                }
            }
        },
        "scope": {
            "project": {
                "name": "xxxxxxxxxxxxxxxxx"
            }
        }
    }
}
```

If all data required for the API request is available, you can send the request to call the API through **curl**, **Postman**, or coding. In the response to the API used to obtain a user token, **x-subject-token** is the desired user token. This token can then be used to authenticate the calling of other APIs.

4.2 Authentication

AK/SK-based Authentication

An AK/SK is used to verify the identity of a request sender. In AK/SK-based authentication, a signature needs to be obtained and then added to requests.

NOTE

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.

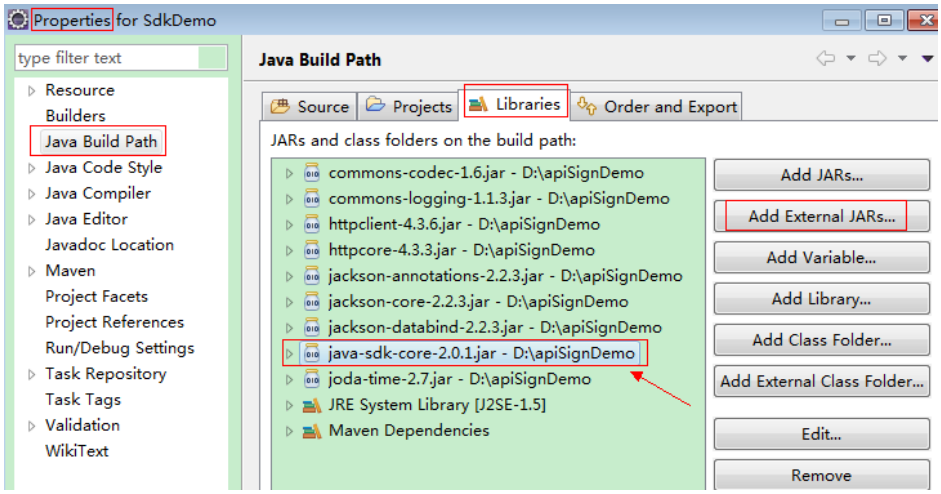
The following uses a demo project to show how to sign a request and use an HTTP client to send an HTTPS request.

Download the demo project at **https://github.com/api-gate-way/SdkDemo**.

If you do not need the demo project, obtain the API Gateway signing SDK from the enterprise administrator.

Decompress the downloaded package and reference the obtained JAR files as dependencies.

Figure 4-1 Importing a JAR file



Step 1 Generate an AK/SK. If an AK/SK pair is already available, skip this step and go to **Step 2**. Find the downloaded AK/SK file, which is usually named **credentials.csv**.

1. Register an account and log in to the management console.
2. Click the username and choose **My Credential** from the drop-down list.
3. On the **My Credentials** page, choose **Access Keys** in the navigation pane.
4. Click **Add Access Key**.
5. Click **OK**. Download the access key after it is created.

   NOTE

   Keep the access key secure.

Step 2 Download and decompress the demo project.

Step 3 Import the demo project to Eclipse.

Figure 4-2 Selecting Existing Projects into Workspace

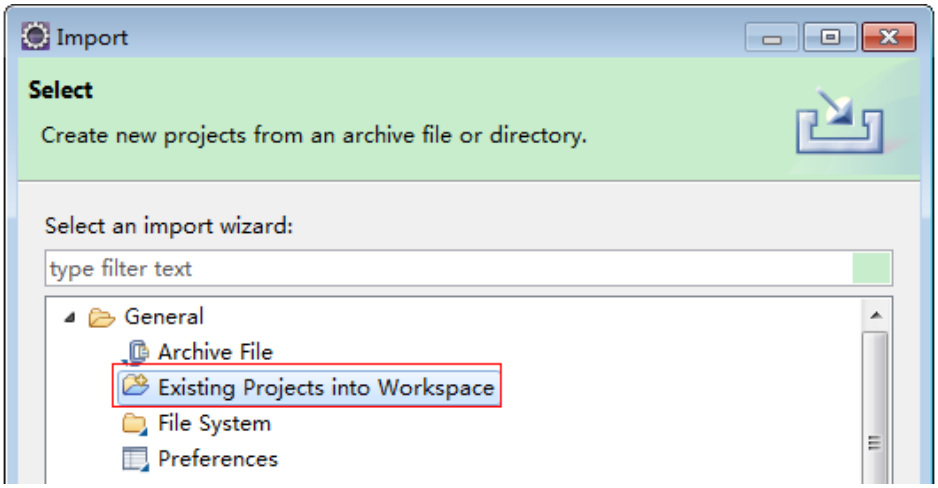


Figure 4-3 Selecting the demo project

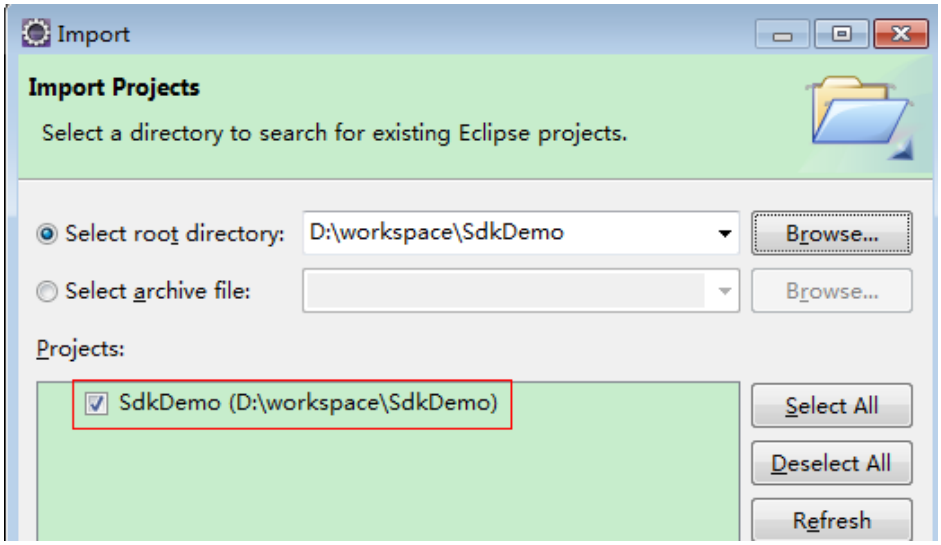
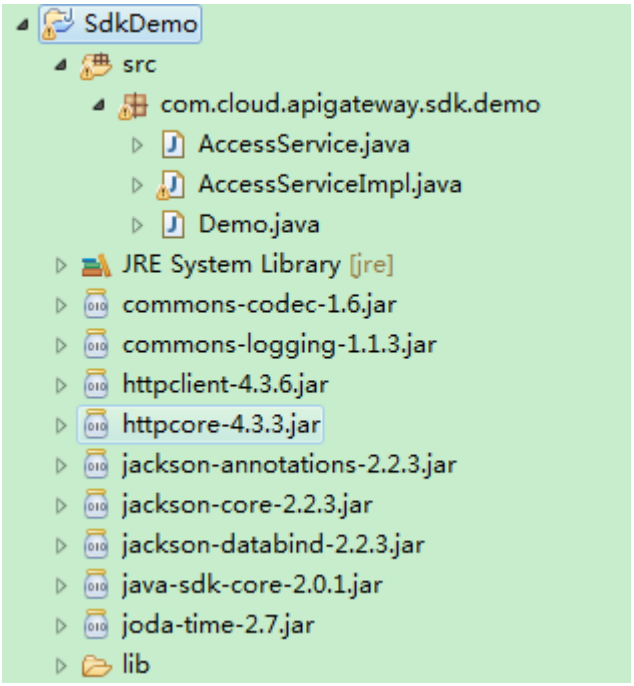


Figure 4-4 Structure of the demo project



Step 4 Sign the request.

The request signing method is integrated in the JAR files imported in **Step 3**. The request needs to be signed before it is sent. The signature will then be added as part of the HTTP header to the request.

The demo code is classified into the following classes to demonstrate signing and sending the HTTP request:

- **AccessService**: An abstract class that merges the GET, POST, PUT, and DELETE methods into the access method.
- **Demo**: Execution entry used to simulate the sending of GET, POST, PUT, and DELETE requests.
- **AccessServiceImpl**: Implements the access method, which contains the code required for communication with API Gateway.

1. Edit the main( ) method in the **Demo.java** file, and replace the bold text with actual values.

   As shown in the following code, if you use other methods such as POST, PUT, and DELETE, see the corresponding comment.

   Specify **region**, **serviceName**, **ak/sk**, and **url** as the actual values. In this demo, the URLs for accessing VPC resources are used.

   To obtain the project ID in the URLs, see **Obtaining a Project ID**. To obtain the endpoint, contact the enterprise administrator.

```
//TODO: Replace region with the name of the region in which the service to be accessed is located.
private static final String region = "";

//TODO: Replace vpc with the name of the service you want to access. For example, ecs, vpc, iam, and elb.
private static final String serviceName = "";

public static void main(String[] args) throws UnsupportedEncodingException
{
```



```
//TODO: Replace the AK and SK with those obtained on the My Credential page.
String ak = System.getenv("CLOUD_SDK_AK")
String sk = System.getenv("CLOUD_SDK_SK")

//TODO: To specify a project ID (multi-project scenarios), add the X-Project-Id header.
//TODO: To access a global service, such as IAM, DNS, CDN, and TMS, add the X-Domain-Id header to specify an account ID.
//TODO: To add a header, find "Add special headers" in the AccessServiceImple.java file.

//TODO: Test the API
String url = "https://{Endpoint}/v1/{project_id}/vpcs";
get(ak, sk, url);

//TODO: When creating a VPC, replace {project_id} in postUrl with the actual value.
//String postUrl = "https://serviceEndpoint/v1/{project_id}/cloudservers";
//String postbody ="{\"vpc\": {\"name\": \"vpc\",\"cidr\": \"192.168.0.0/16\"}}";
//post(ak, sk, postUrl, postbody);

//TODO: When querying a VPC, replace {project_id} in url with the actual value.
//String url = "https://serviceEndpoint/v1/{project_id}/vpcs/{vpc_id}";
//get(ak, sk, url);

//TODO: When updating a VPC, replace {project_id} and {vpc_id} in putUrl with the actual values.
//String putUrl = "https://serviceEndpoint/v1/{project_id}/vpcs/{vpc_id}";
//String putbody ="{\"vpc\":{\"name\": \"vpc1\",\"cidr\": \"192.168.0.0/16\"}}";
//put(ak, sk, putUrl, putbody);

//TODO: When deleting a VPC, replace {project_id} and {vpc_id} in deleteUrl with the actual values.
//String deleteUrl = "https://serviceEndpoint/v1/{project_id}/vpcs/{vpc_id}";
//delete(ak, sk, deleteUrl);
}
```

2. Compile the code and call the API.

   In the **Package Explorer** area on the left, right-click **Demo.java**, choose **Run AS** > **Java Application** from the shortcut menu to run the demo code.

   You can view the API call logs on the console.

----End

4.3 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 **Status Code**.

For example, if status code **201** is returned for calling the API used to obtain a user token, the request is successful.

Response Header

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

The following shows the response header for the API to obtain a user token, in which **x-subject-token** is the desired user token. This token can then be used to authenticate the calling of other APIs.

Figure 4-5 Header fields of the response to the request for obtaining a user token

(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.

The following shows part of the response body for the API to obtain a user token. For the sake of space, only part of the content is displayed here.

```
{
    "token": {
        "expires_at": "2019-02-13T06:52:13.855000Z",
        "methods": [
            "password"
        ],
        "catalog": [
            {
                "endpoints": [
                    {
                        "region_id": "xxxxxxxx",
.....
```

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": "The format of message is error",
    "error_code": "AS.0001"
}
```

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

4.4 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 **Status Code**.

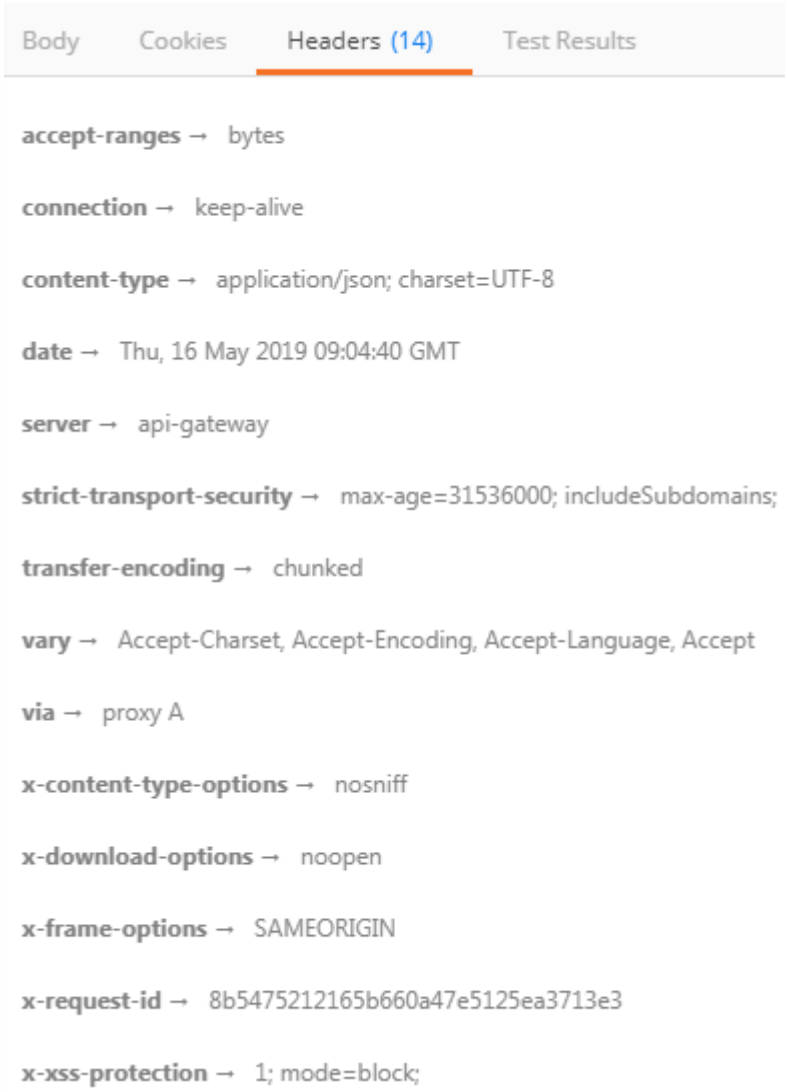
For example, if status code **200** is returned for calling the API used to obtain a VPC list, the request is successful.

Response Header

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

Figure 4-6 shows the response header fields for the API used to obtain a VPC list.

Figure 4-6 Header fields of the response to the request for obtaining a VPC list



(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.

The following shows the response body for the API used to obtain a VPC list. In the response body, the VPC named **vpc-test** is displayed.

```
{
   "vpcs": [
      {
         "id": "7e1a7e70-3f3e-4581-955e-26a4848d8f31",
         "name": "vpc-test",
         "description": "",
         "cidr": "192.168.0.0/16",
         "status": "OK",
         "routes": [],
         "enterprise_project_id": "0"
      }
   ]
}
```


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

```
{
    "code": "VPC.0002",
    "message": "Available zone Name is null."
}
```

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

5 API

5.1 Anti-DDoS Management

5.1.1 Querying the List of Defense Statuses of EIPs

Function

Querying the List of Defense Statuses of EIPs

URI

GET /v1/{project_id}/antiddos

Table 5-1 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**

Table 5-2 Query Parameters

Parameter	Mandatory	Type	Description
status	No	String	status
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.

Parameter	Mandatory	Type	Description
ip	No	String	IP address. Both IPv4 and IPv6 addresses are supported. For example, if you enter ?ip=192.168, the defense status of EIPs corresponding to 192.168.111.1 and 10.192.168.8 is returned.

Request Parameters

Table 5-3 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Default: **application/json;charset=utf8**

Response Parameters

Status code: 200

Table 5-4 Response body parameters

Parameter	Type	Description
total	Integer	total count
ddosStatus	Array of **DDosStatus** objects	Anti-DDoS status

Table 5-5 DDosStatus

Parameter	Type	Description
floating_ip_id	String	ID of an EIP

Parameter	Type	Description
floating_ip_address	String	IP address. Both IPv4 and IPv6 addresses are supported. For example, if you enter ?ip=192.168, the defense status of EIPs corresponding to 192.168.111.1 and 10.192.168.8 is returned.
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	Defense status
blackhole_endtime	Long	The end time of blackhole
protect_type	String	Protect type
traffic_threshold	Long	Threshold of traffic
http_threshold	Long	Threshold of http traffic

Example Requests

None

Example Responses

Status code: 200

OK

```
{
  "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
  } ]
}
```

Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

5.1.2 Querying Optional Anti-DDoS Defense Policies

Function

Querying Optional Anti-DDoS Defense Policies

URI

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

Table 5-6 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**

Request Parameters

Table 5-7 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Default: **application/json;charset=utf8**

Response Parameters

Status code: 200

Table 5-8 Response body parameters

Parameter	Type	Description
traffic_limited_list	Array of **TriggerBpsDict** objects	List of traffic limits

Parameter	Type	Description
http_limited_list	Array of **TriggerQpsDict** objects	List of HTTP limits
connection_limited_list	Array of **CleanLimitDict** objects	List of limits of numbers of connections
extend_ddos_config	Array of **ExtendDDoSSet** objects	List of extend ddos limits

Table 5-9 TriggerBpsDict

Parameter	Type	Description
traffic_pos_id	Long	Position ID of traffic
traffic_per_second	Long	Threshold of traffic per second (Mbit/s)
packet_per_second	Long	Threshold of number of packets per second

Table 5-10 TriggerQpsDict

Parameter	Type	Description
http_request_pos_id	Long	Position ID of number of HTTP requests
http_packet_per_second	Long	Threshold of number of HTTP requests per second

Table 5-11 CleanLimitDict

Parameter	Type	Description
cleaning_access_pos_id	Long	Position ID of access limit during cleaning
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 5-12 ExtendDDoSSet

Parameter	Type	Description
SetID	Long	Position ID of config
new_connection_limited	Long	List of limits of numbers of connections
total_connection_limited	Long	List of limits of numbers of total connections
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

OK


```
{
  "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" : [ ]
}
```


Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

5.1.3 Querying Weekly Defense Statistics

Function

Querying Weekly Defense Statistics

URI

GET /v1/{project_id}/antiddos/weekly

Table 5-13 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**

Table 5-14 Query Parameters

Parameter	Mandatory	Type	Description
period_start_date	No	String	Start date of a seven-day period

Request Parameters

Table 5-15 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Content-Type Default: **application/json;charset=utf8**

Response Parameters

Status code: 200

Table 5-16 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 5-17 WeeklyCount

Parameter	Type	Description
ddos_intercept_times	Integer	Number of DDoS attacks blocked

Parameter	Type	Description
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 date

Table 5-18 WeeklyTop10

Parameter	Type	Description
floating_ip_address	String	EIP
times	Integer	Number of DDoS attacks intercepted, including cleaning operations and black holes

Example Requests

None

Example Responses

Status code: 200

OK

```
{
  "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" : [ ]
}
```


Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

5.1.4 Querying Configured Anti-DDoS Defense Policies

Function

Querying Configured Anti-DDoS Defense Policies

URI

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

Table 5-19 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**
floating_ip_id	Yes	String	ID corresponding to the EIP of a user

Table 5-20 Query Parameters

Parameter	Mandatory	Type	Description
ip	No	String	IP address.

Request Parameters

Table 5-21 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Content-Type Default: **application/json;charset=utf8**

Response Parameters

Status code: 200

Table 5-22 Response body parameters

Parameter	Type	Description
enable_L7	Boolean	Whether L7 defense has been enabled
traffic_pos_id	Long	Position ID of traffic. The value ranges from 1 to 9 and 33 to 36.
http_request_pos_id	Long	Position ID of number of HTTP requests. The value ranges from 1 to 15 and 33 to 36.
cleaning_access_pos_id	Long	Position ID of access limit during cleaning. The value ranges from 1 to 8 and 33 to 36.
app_type_id	Long	Application type ID.

Example Requests

None

Example Responses

Status code: 200

OK

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

Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

5.1.5 Updating Anti-DDoS Defense Policies

Function

Updating Anti-DDoS Defense Policies

URI

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

Table 5-23 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**
floating_ip_id	Yes	String	ID corresponding to the EIP of a user

Table 5-24 Query Parameters

Parameter	Mandatory	Type	Description
ip	No	String	ip

Request Parameters

Table 5-25 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Default: **application/json;charset=utf8**

Table 5-26 Request body parameters

Parameter	Mandatory	Type	Description
app_type_id	Yes	Integer	Application type ID.
cleaning_access_pos_id	Yes	Integer	Position ID of access limit during cleaning. The value ranges from 1 to 8 and 33 to 36.
enable_L7	Yes	Boolean	Whether L7 defense has been enabled
http_request_pos_id	Yes	Integer	Position ID of number of HTTP requests. The value ranges from 1 to 15 and 33 to 36.
traffic_pos_id	Yes	Integer	Position ID of traffic. The value ranges from 1 to 9 and 33 to 36.

Response Parameters

Status code: 200

Table 5-27 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.

Example Requests

```
{
  "app_type_id" : 0,
  "cleaning_access_pos_id" : 8,
  "enable_L7" : false,
  "http_request_pos_id" : 1,
  "traffic_pos_id" : 1
}
```

Example Responses

Status code: 200

OK

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

Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

5.1.6 Querying the Traffic of a Specified EIP

Function

Querying the Traffic of a Specified EIP

URI

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

Table 5-28 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**
floating_ip_id	Yes	String	ID corresponding to the EIP of a user

Table 5-29 Query Parameters

Parameter	Mandatory	Type	Description
ip	No	String	IP address. Both IPv4 and IPv6 addresses are supported. For example, if you enter ?ip=192.168, the defense status of EIPs corresponding to 192.168.111.1 and 10.192.168.8 is returned.

Request Parameters

Table 5-30 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Content-Type Default: **application/json;charset=utf8**

Response Parameters

Status code: 200

Table 5-31 Response body parameters

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

Table 5-32 DailyData

Parameter	Type	Description
period_start	Long	Start time
bps_in	Integer	Inbound traffic (bit/s)
bps_attack	Long	Attack traffic (bit/s)

Parameter	Type	Description
total_bps	Long	Total traffic
pps_in	Long	Inbound packet rate (number of packets per second)
pps_attack	Long	Attack packet rate (number of packets per second)
total_pps	Long	Total packet rate

Example Requests

None

Example Responses

Status code: 200

OK

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

Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

5.1.7 Querying Events of a Specified EIP

Function

Querying Events of a Specified EIP

URI

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

Table 5-33 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**
floating_ip_id	Yes	String	ID corresponding to the EIP of a user

Table 5-34 Query Parameters

Parameter	Mandatory	Type	Description
sort_dir	No	String	Possible values: desc: indicates that query results are given and sorted by time in descending order. asc: indicates that query results are given and sorted by time in ascending order.
limit	No	String	Limit of number of returned results or the maximum number of returned results of a query.
offset	No	String	Offset. This parameter is valid only when used together with the limit parameter.
ip	No	String	IP address. Both IPv4 and IPv6 addresses are supported. For example, if you enter ? ip=192.168, the defense status of EIPs corresponding to 192.168.111.1 and 10.192.168.8 is returned.

Request Parameters

Table 5-35 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Content-Type Default: **application/json;charset=utf8**

Response Parameters

Status code: 200

Table 5-36 Response body parameters

Parameter	Type	Description
total	Long	Total number of EIPs
logs	Array of **DailyLog** objects	List of events

Table 5-37 DailyLog

Parameter	Type	Description
start_time	Long	Start time
end_time	Long	End time
status	Integer	Defense status
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

OK

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

Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

5.1.8 Querying Configured Anti-DDoS Defense Policies

Function

Querying Configured Anti-DDoS Defense Policies

URI

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

Table 5-38 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**
floating_ip_id	Yes	String	ID corresponding to the EIP of a user

Table 5-39 Query Parameters

Parameter	Mandatory	Type	Description
ip	No	String	IP address. Both IPv4 and IPv6 addresses are supported. For example, if you enter ?ip=192.168, the defense status of EIPs corresponding to 192.168.111.1 and 10.192.168.8 is returned.

Request Parameters

Table 5-40 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Content-Type Default: **application/json;charset=utf8**

Response Parameters

Status code: 200

Table 5-41 Response body parameters

Parameter	Type	Description
status	String	Ddos status

Example Requests

None

Example Responses

Status code: 200

OK

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


Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

5.2 Anti-DDoS Task Management

5.2.1 Querying Anti-DDoS Tasks

Function

Querying Anti-DDoS Tasks

URI

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

Table 5-42 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**

Table 5-43 Query Parameters

Parameter	Mandatory	Type	Description
task_id	Yes	String	Task ID (nonnegative integer) character string

Request Parameters

Table 5-44 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Content-Type Default: **application/json;charset=utf8**

Response Parameters

Status code: 200

Table 5-45 Response body parameters

Parameter	Type	Description
task_status	String	Status of a task
task_msg	String	Additional information about a task

Example Requests

None

Example Responses

Status code: 200

OK

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

Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

5.3 Alarm configuration management

5.3.1 Querying alert config

Function

Querying alert config

URI

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

Table 5-46 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**

Request Parameters

Table 5-47 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Content-Type Default: **application/json;charset=utf8**

Response Parameters

Status code: 200

Table 5-48 Response body parameters

Parameter	Type	Description
topic_urn	String	ID of an alarm group
display_name	String	display name of alert config
warn_config	**warn_config** object	warn config

Table 5-49 warn_config

Parameter	Type	Description
antiDDoS	Boolean	DDoS attacks
back_doors	Boolean	Webshells
bruce_force	Boolean	Brute force cracking (system logins, FTP, and DB)
high_privilege	Boolean	Overly high rights of a database process
remote_login	Boolean	Alarms about remote logins
send_frequency	Integer	Possible values:0: indicates that alarms are sent once a day.1: indicates that alarms are sent once every half hour.This parameter is mandatory for the Host Intrusion Detection (HID) service.
waf	Boolean	Reserved
weak_password	Boolean	Weak passwords (system and database)

Example Requests

None

Example Responses

Status code: 200

OK

```
{
  "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
}
```

Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

5.3.2 Configuring alert config

Function

Configuring alert config

URI

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

Table 5-50 Path Parameters

Parameter	Mandatory	Type	Description
project_id	Yes	String	poject id Minimum: **32** Maximum: **64**

Request Parameters

Table 5-51 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	token Minimum: **32** Maximum: **2097152**
Content-Type	Yes	String	Content-Type Default: **application/json;charset=utf8**

Table 5-52 Request body parameters

Parameter	Mandatory	Type	Description
display_name	Yes	String	display name of alert config
topic_urn	Yes	String	topic urn
warn_config	Yes	**WarnConfig** object	Alarm configuration information.

Table 5-53 WarnConfig

Parameter	Mandatory	Type	Description
antiDDoS	Yes	Boolean	DDoS attack
back_doors	No	Boolean	Web page backdoor
bruce_force	No	Boolean	Brute force cracking (system login, FTP, database)
high_privilege	No	Boolean	The database process permission is too high.
remote_login	No	Boolean	Remote login reminder
send_frequency	No	Integer	Value range: • 0: once a day • 1: once a half an hour This parameter is mandatory for HIDs.
waf	No	Boolean	Reserved field.
weak_password	No	Boolean	Weak password (system, database)

Response Parameters

Status code: 200

Table 5-54 Response body parameters

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

Example Requests

```
{
  "display_name" : "",
  "topic_urn" : "8",
  "warn_config" : {
    "antiDDoS" : false,
    "back_doors" : false,
    "bruce_force" : false,
    "high_privilege" : false,
    "remote_login" : false,
    "send_frequency" : 10,
    "waf" : false,
    "weak_password" : false
  }
}
```

Example Responses

Status code: 200

OK

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

Status Codes

Status Code	Description
200	OK

Error Codes

See **Error Codes**.

6 Examples

6.1 Example 1: Updating Defense Policy for an IP Address

Scenario

You can modify the defense policy configured for an IP address on the Anti-DDoS console or by calling the API.

Process:

1. Query the defense statuses of all IP addresses.
2. Query the optional Anti-DDoS defense policies.
3. Update the defense policy for an IP address.
4. Query the execution status of the defense policy updating task based on the task ID returned in **3**.

Involved APIs

The following APIs are required for updating the Anti-DDoS defense policy for an IP address:

- **Step 1** is used for querying the defense statuses of all IP addresses.
- **Step 2** is used for querying the optional Anti-DDoS defense policies.
- **Step 3** is used for updating the defense policy for an IP address.
- **Step 4** is used for querying the execution status of the defense policy updating task based on the task ID.

Procedure

Step 1 Query the defense statuses of all IP addresses.

- API information

  URI format: **GET /v1/{*project_id*}/antiddos**

  For details, see section "Querying the List of Defense Statuses of EIPs".

- Example request

  GET: https://*{endpoint}*/v1/1858a4e1f99d4454bd6a539d5477f5de/antiddos

  Obtain *{endpoint}* from .

  Body:

  ```
  {
  }
  ```

- Example response

  ```
  {
  "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
  }
  ]
  }
  ```

Step 2 Query the optional Anti-DDoS defense policies.

- API information

  URI format: **GET /v2/{*project_id*}/antiddos/query-config-list**

  For details, see section "Querying Optional Anti-DDoS Defense Policies".

- Example request

  GET: https://*{endpoint}*/v2/1858a4e1f99d4454bd6a539d5477f5de/antiddos/query-config-list

  Obtain *{endpoint}* from .

  Body:

  ```
  {
  }
  ```

- Example response

  ```
  {
  "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": []
}
```


Step 3 Update the defense policy for an IP address.

- API information

URI format: **PUT /v1/{*project_id*}/antiddos/{floating_ip_id}**

For details, see section "Updating Anti-DDoS Defense Policies".

- Example request

  PUT: https://*{endpoint}*/v1/1858a4e1f99d4454bd6a539d5477f5de/antiddos/18e6ace5-eb36-4196-a15e-1e000c24e026

  Obtain *{endpoint}* from .

  Body:

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

- Example response

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

Step 4 Query the execution status of the defense policy updating task based on the task ID returned in **Step 3**.

- API information

  URI format: **GET /v2/{*project_id*}/query-task-status**

  For details, see section "Querying Anti-DDoS Tasks".

- Example request

  GET: https://*{endpoint}*/v2/1858a4e1f99d4454bd6a539d5477f5de/query-task-status?task_id=59385d2a-6266-4d3a-9122-a228c530f557

  Obtain *{endpoint}* from .

  Body:

  ```
  {
  }
  ```

- Example response

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

----End

6.2 Example 2: Configuring the Default Protection Policy

Scenario

You can configure the default protection policy for newly purchased IP addresses on the Anti-DDoS console or by call the API.

Process:

1. Query the default protection settings for the newly purchased IP addresses.
2. Configure the default protection policy for newly purchased IP addresses.

Involved APIs

The following APIs are required for configuring default protection policy for newly purchased IP addresses.

- **Step 1** is used to query default protection settings for the newly purchased IP addresses.
- **Step 2** is used to configure default protection policy for newly purchased IP addresses.

Procedure

Step 1 Query default protection settings for the newly purchased IP addresses.

- API information

  URI format: **GET /v1/{*project_id*}/antiddos/default-config**

  For details, see section "Querying the Default Protection Policy Configured for the Newly Purchased Public IP Addresses".

- Example request

  GET: https://*{endpoint}*/v1/1858a4e1f99d4454bd6a539d5477f5de/antiddos/default/config

  Obtain *{endpoint}* from .

  Body:

  ```
  {
  }
  ```

- Example response

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

Step 2 Configure default protection policy for newly purchased IP addresses.

- API information

  URI format: **POST /v1/{*project_id*}/antiddos/default-config**

  For details, see section "Configuring the Default Protection Policy for Newly Purchased Public IP Addresses".

- Example request

  POST: https://*{endpoint}*/v1/1858a4e1f99d4454bd6a539d5477f5de/antiddos/default-config

  Obtain *{endpoint}* from .

  Body:

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



```
"traffic_pos_id": 8
}
```

- Example response

```
{
}
```

----End

A Appendix

A.1 Status Code

- 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 Error Codes

Status Code	Error Codes	Error Message	Description	Solution
200	Anti-DDoS.0	Succeeded	Succeeded	No need to deal with it
200	Anti-DDoS.10000000	The task has been received and is being handled	The task has been received and is being handled	No need to deal with it
400	Anti-DDoS.10000001	Enter a valid request message	Enter a valid request message	Check the parameters
400	Anti-DDoS.10001008	An incorrect task ID is used	An incorrect task ID is used	Check the parameters
400	Anti-DDoS.10001010	Invalid time	Invalid time	Check the parameters
401	Anti-DDoS.10000004	Public test service denied	Public test service denied	Apply for public test

Status Code	Error Codes	Error Message	Description	Solution
403	Anti-DDoS.10000002	Failed to authenticate the token in the request	Failed to authenticate the token in the request	Re apply for token
403	Anti-DDoS.10000009	The account is restricted	The account is restricted	Apply for authority
403	Anti-DDoS.10000010	The account is frozen	The account is frozen	Apply for unfreeze
403	Anti-DDoS.10000012	Unknown user type	Unknown user type	Apply for authority
403	Anti-DDoS.10000016	VPC access failed or EIP is not exist	VPC access failed or EIP is not exist	Contact to administrator
403	Anti-DDoS.10000030	You have not been authenticated. Perform real-name authentication first.	You have not been authenticated. Perform real-name authentication first.	Real name authentication
403	Anti-DDoS.10001009	The operation permission is restricted	The operation permission is restricted	Apply for authority
403	Anti-DDoS.11000001	Access to the database is rejected	Access to the database is rejected	Contact to administrator
500	Anti-DDoS.11000000	Internal system exception. Contact technical support engineers	Internal system exception. Contact technical support engineers	Contact to administrator

A.3 Obtaining a Project ID

Obtaining a Project ID by Calling an API

You can obtain the project ID by calling the IAM API used to query project information based on the 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 the administrator. 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 Credential** from the drop-down list.

   On the **My Credential** page, view project IDs in the project list.

B Out-of-Date APIs

B.1 Querying Optional Anti-DDoS Defense Policies

NOTE

This API is a historical API and may not be maintained in the future. Please use the API in section "Querying Optional Anti-DDoS Defense Policies".

Functions

This API allows you to query optional Anti-DDoS defense policies. Based on your service, you can select a policy for Anti-DDoS traffic cleaning.

URI

- URI format

  GET /v1/{project_id}/antiddos/query_config_list
- Parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	User ID

Request

Example request

```
GET /v1/67641fe6886f43fcb78edbbf0ad0b99f/antiddos/query_config_list
```

Response

- Parameter description

Parameter	Mandatory	Type	Description
traffic_limited_list	Yes	List data structure	List of traffic limits
http_limited_list	Yes	List data structure	List of HTTP limits
connection_limited_list	Yes	List data structure	List of limits of numbers of connections

- Data structure description of **traffic_limited_list**

Parameter	Mandatory	Type	Description
traffic_pos_id	Yes	Integer	Position ID of traffic
traffic_per_second	Yes	Integer	Threshold of traffic per second (Mbit/s)
packet_per_second	Yes	Integer	Threshold of number of packets per second

- Data structure description of **http_limited_list**

Parameter	Mandatory	Type	Description
http_request_pos_id	Yes	Integer	Position ID of number of HTTP requests
http_packet_per_second	Yes	Integer	Threshold of number of HTTP requests per second

- Data structure description of **connection_limited_list**

Parameter	Mandatory	Type	Description
cleaning_access_pos_id	Yes	Integer	Position ID of access limit during cleaning
new_connection_limited	Yes	Integer	Number of new connections of a source IP address
total_connection_limited	Yes	Integer	Total number of connections of a source IP address

- Example response

```
{
  "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
  }
 ],
 "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": [
    {
      "new_connection_limited": 80,
      "total_connection_limited": 700,
      "http_packet_per_second": 500000,
      "traffic_per_second": 1000,
      "packet_per_second": 200000,
      "setID": 33
    },
    {
      "new_connection_limited": 80,
      "total_connection_limited": 700,
      "http_packet_per_second": 500000,
      "traffic_per_second": 2000,
      "packet_per_second": 200000,
      "setID": 34
    },
    {
      "new_connection_limited": 80,
      "total_connection_limited": 700,
      "http_packet_per_second": 500000,
      "traffic_per_second": 5000,
      "packet_per_second": 400000,
      "setID": 35
    },
    {
      "new_connection_limited": 80,
      "total_connection_limited": 700,
      "http_packet_per_second": 0,
      "traffic_per_second": 0,
      "packet_per_second": 0,
      "setID": 36
    }
  ]
}
```


NOTE

The **extend_ddos_config** field displays information about Anti-DDoS defense policies set by users based on their needs.

Status Code

For details, see **Status Code**.

B.2 Querying Anti-DDoS Tasks

NOTE

This API is a historical API and may not be maintained in the future. Please use the API in section "Querying Anti-DDoS Tasks".

Functions

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

URI

- URI format

  GET /v1/{project_id}/query_task_status

  NOTE

  You can use **?** and **&** behind the URI to add query conditions, as shown in the request example.

- Parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	User ID

Request

- Parameter description

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

- Example request

```
GET /v1/67641fe6886f43fcb78edbbf0ad0b99f/query_task_status?
task_id=4a4fefe7-34a1-40e2-a87c-16932af3ac4a
```

Response

- Parameter description

Name	Type	Description
task_status	String	Status of a task, which can be one of the following: • success • failed • waiting • running • preprocess • ready
task_msg	String	Additional information about a task

- Example response


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

Status Code

For details, see **Status Code**.

