Application Performance Management

API Reference

Issue 02

Date 2022-06-20

HUAWEI TECHNOLOGIES CO., LTD.

Contents

1 API Overview

Application Performance Management (APM) provides APIs for developers and partners to quickly implement application O&M at low costs.

APM supports RESTful APIs. Representational State Transfer (REST) allocates Uniform Resource Identifiers (URIs) to dispersed resources so that the resources can be located. Applications on clients use Unified Resource Locators (URLs) to obtain the resources.

A URL is usually in the format of "https://*Endpoint*/*uri*". *uri* indicates the resource path, that is, the API access path.

Table 1-1 URL parameters

Parameter	Description
Endpoint	Endpoints in the URL vary according to services. For details about endpoints of APM, see **Table 1-2**.
uri	Resource path, that is, the API access path. The path can be obtained from the URI module of an API. For example, for the API for querying the service list (**URI**), the path is **/v1/{*project_id*}/ats/applications**. {*project_id*} indicates the project ID. For details, see **Example**.

Table 1-2 Endpoint of APM

Service Name	Region Name	Region	Endpoint	Protocol
APM	CN-Hong Kong	ap-southeast-1	apm.ap-southeast-1.myhuaweicloud.com	HTTPS

2 Using APIs

2.1 Calling APIs

Before APIs are called, request authentication information needs to be obtained and padded in the request header. Currently, only token-based authentication is supported.

The process for calling an APM API is as follows: constructing requests > initiating requests > parsing responses.

The following lists the steps for calling an API:

1. **Obtaining Information for Authentication**
2. **Constructing Requests**
3. **Initiating Requests**
4. **Parsing Responses**

2.2 Token-based Authentication

Application Scenario

For token-based authentication, you must obtain your token and add **X-Auth-Token** to the headers of API requests.

This section describes how to authorize API requests using tokens. For details about how to obtain a token, see **Obtaining a User Token Through Password Authentication**.

Procedure

1. Send **POST https://*Endpoint of IAM*/v3/auth/tokens** to obtain the endpoint of Identity and Access Management (IAM) and the region name in the message body. For details, see **Endpoint of IAM**.

Table 2-1 Endpoint of IAM

Service Name	Region Name	Region	Endpoint	Protocol
IAM	CN-Hong Kong	ap-southeast-1	iam.ap-southeast-1.myhuaweicloud.com	HTTPS

Example request:

```
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "name": "username",//Replace the username as required.
          "password": "password",//Replace the password as required.
          "domain": {
            "name": "domainname"//Replace the domainname as required.
          }
        }
      }
    },
    "scope": {
      "project": {
        "name": "ap-southeast-1" //ap-southeast-1 indicates the IAM region. Replace it as required.

      }
    }
  }
}
```

2. Obtain a token. After a response is returned, the value of **X-Subject-Token** in the response header is the token value.
3. Call a service API and add **X-Auth-Token** to the request header. The value of **X-Auth-Token** is the token value obtained in 2.

2.3 Constructing Requests

A request consists of three parts: a request line, request header, and request body (optional).

Request Line

A request line consists of three parts in order, separated by spaces: a method token, a request URI, and a protocol version. The format is shown in the following code:

```
Method Request-URI HTTP-Version CRLF
```

- Method: indicates a request method. All methods are capitalized. Their meanings are as follows:
  – GET: obtains the resource identified by a Request-URI.

  - POST: suffixes new data to the resource identified by a Request-URI.
  - PUT: requests a server to store a resource and uses a Request-URI to identify the resource.
  - DELETE: requests the server to delete the resource identified by a Request-URI.
  - PATCH: requests a server to update a resource or create a resource if the target resource does not exist.
- Request-URI: indicates a unified resource identifier.

  NOTE

  Enter a question mark (?) and an ampersand (&) at the end of the URI to define multiple search criteria. The content contained in **{}** in the URI is the parameter of the URI, where **?** is contained. The previous part is the path parameter, and the latter part is the query parameter.
- HTTP-Version: indicates the version of the HTTP protocol used by a request.
- CRLF: indicates the carriage return and line feed characters. CRLF is placed only at the end of a line, and a separate CR or LF is not allowed.

Request Header

A request header consists of several header fields. Each header field consists of a domain name, a colon (:), and a field value.

Common request headers are listed in **Table 5-3**.

Request Body

A request message body is a JSON-style nested key-value pair. The mandatory and optional fields of an HTTP request body vary depending on the URI.

2.4 Initiating Requests

There are two ways to initiate requests:

- cURL

  cURL is a command line tool used to perform URL operations and transmit information. It serves as a client to send HTTP requests to a server. You can use cURL to initiate requests for API commissioning.
- Code

  You can call APIs using code to assemble, send, and process requests.

2.5 Parsing Responses

After receiving and interpreting a request message, the server returns an HTTP response message.

A response also consists of three parts: a status line, a response header, and a response body.

Status Line

The format of the status line is as follows:

```
HTTP-Version Status-Code Reason-Phrase CRLF
```

- **HTTP-Version** indicates the version of the HTTP protocol used by the server.
- **Status-Code** indicates the status code in the response returned by the server.

  A status code consists of three digits. The first digit defines the category of a response. There are five possible values for the first digit:
  - 1xx: informational response, indicating that the request has been received and the recipient is continuing to process the request.
  - 2xx: success response, indicating that the request has been received, understood, and accepted.
  - 3xx: redirection response, indicating that further actions need to be taken to complete the request.
  - 4xx: client error response, indicating that the request contains a syntax error or cannot be fulfilled.
  - 5xx: server error response, indicating that the server has failed to fulfill a valid request.
- **Reason-Phrase** indicates the text description of a status code.

Response Header

For details about common response headers, see **Table 5-4**.

Response Body

A response body is a JSON text. If an error occurs in calling an API, the server returns a response body containing an error code and error description.

2.6 Example

This section describes how to call the API for querying the service list. In the following example, instances are deployed on **SpringCloudDemo**. Spring Cloud is a microservice architecture development tool based on Spring Boot. It facilitates configuration management, service governance, circuit breakers, intelligent routing, micro-proxy, control bus, global locks, leadership election, distributed sessions, and cluster state management involved in the microservice architecture.

To call the API used to query the service list, do as follows:

1. Obtain the token, which will be put into the request header for authentication in a subsequent request.
2. Construct and initiate a request to call the API used to query the service list.
3. View the obtained response, and check whether the API used to query the service list is successfully called.

Preparations:

1. The **SpringCloudDemo** application has been deployed on the server.

2. The endpoints of IAM and APM have been obtained. In this example, the endpoint of IAM is **iam_Endpoint**, and the endpoint of APM is **apm_Endpoint**.

Step 1 Obtain the token for authentication.

1. Enter **POST https://***IAM endpoint***/v3/auth/tokens**. In this example, *IAM endpoint* is **iam_Endpoint**. Therefore, enter **POST https://iam_Endpoint/v3/auth/tokens**.
2. After a response is returned, the value of **X-Subject-Token** in the response header is the token value.

   Example request:

```
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "name": "username",//Replace the username as required.
          "password": "password",//Replace the password as required.
          "domain": {
            "name": "domainname"//Replace the domainname as required.
          }
        }
      }
    },
    "scope": {
      "project": {
        "name": "ap-southeast-1" //ap-southeast-1 indicates the IAM region. Replace it as required.

      }
    }
  }
}
```

Step 2 Construct a request and send it.

A request consists of three parts: a request line, request header, and request body (optional). Construct a request, send it, and wait for the response.

1. Request line

   GET https://*Endpoint*/v1/*{project_id}*/ats/applications

   Endpoint indicates the endpoint of APM, which is **apm_Endpoint** in this example.

   {project_id} indicates the path parameter.

Parameter	Type	Description
project_id	String	Project ID.

During token authentication, the response body contains the value of **{project_id}**, which is **id: 12ff18574dfe4b92......** in this example.

Request line:

https://***apm_Endpoint***/v1/***12ff18574dfe4b92......***/ats/applications?monitorGroup=***SpringCloudDemo***

2. Request header

   X-Auth-Token. The value is the value of **X-subject-token** obtained in token authentication.

   X-Auth-Token: **MIIRpQYJKoZIhvcNAQcCoIIRljCCEZICAQExDTALBglghkgBZQMEAgEwgg-zBgkqhkiG9w0BBwGggg...**

   Content-Type: **application/json;charset=utf8**

3. Request body (not involved in this example)

Step 3 Receive the response.

After the request is sent, a response is returned.

Response:

```
{
  "errorCode": "SVCSTG.ATS.2000",
  "errorMessage": null,
  "responseInfo": [
    "spring-cloud-service-eureka-server",
    "spring-cloud-testservice-consumer",
    "spring-cloud-testservice-provider",
    "springboot1.2",
    "springboot1.3.8",
    "springboot1.4.7",
    "springboot1.5.13"
  ]
}
```

The response consists of **errorCode**, **errorMessage**, and **responseInfo**.

responseInfo indicates all the services that are queried on **SpringCloudDemo**.

----End

3 APIs

3.1 Querying the Application List

Function

This API is used to query the application list.

URI

GET /v1/{project_id}/atps/monitorgroups

Request

Table 3-1 Path parameter

Parameter	Type	Description
project_id	String	Project ID.

Example request

```
/v1/0/atps/monitorgroups
```

Response

Table 3-2 Response parameters

Parameter	Type	Description
errorCode	String	Error code. SVCSTG.ATPS.2000: Query succeeded.
errorMessage	String	Description of the error.

Parameter	Type	Description
responseInfo	List (string)	Application name list.

Example response

```
{
  "errorCode": "SVCSTG.ATPS.2000",
  "errorMessage":null,
  "responseInfo": ["11d5c9b83c1b2e04579fa5a34d191bb5"]
}
```

Status Code

- Success response

  Table 3-3 describes the status code.

  Table 3-3 Status code

Status Code	Description
200	The request is successful.

3.2 Querying the Service List

Function

This API is used to query the service list.

URI

GET /v1/{project_id}/ats/applications

Request

Table 3-4 Request headers

Parameter	Description	Mandatory	Example
clusterId	Cluster ID.	No	default
namespace	Namespace.	No	manage

Table 3-5 Path parameter

Parameter	Type	Description
project_id	String	Project ID.

Table 3-6 Request parameter

Parameter	Mandatory	Type	Description
monitorGroup	No	String	Application name. For details, see **Querying the Application List**.

Example request

```
/v1/0/ats/applications?monitorGroup=11d5c9b83c1b2e04579fa5a34d191bb5
```

Response

Table 3-7 Response parameters

Parameter	Type	Description
errorCode	String	Error code. SVCSTG.ATS.2000: Query succeeded. SVCSTG.ATS.400101: Parameter verification failed. SVCSTG.ATS.200103: No service information found.
errorMessage	String	Description of the error.
responseInfo	List (string)	Service name list.

Example response

```
{
  "errorCode": "SVCSTG.ATS.2000",
  "errorMessage":null,
  "responseInfo": ["ams-calc:8080","ams-metric:8080"]
}
```

Status Code

- Success response

  Table 3-8 describes the status code.

Table 3-8 Status code

Status Code	Description
200	The request is successful.

3.3 Querying the Service Instance List

Function

This API is used to query the instance list of a specified service.

URI

GET /v1/{project_id}/ats/applications/{application}/instances

Request

Table 3-9 Request headers

Parameter	Description	Mandatory	Example
clusterId	Cluster ID.	No	default
namespace	Namespace.	No	manage

Table 3-10 Path parameters

Parameter	Type	Description
project_id	String	Project ID.
application	String	Service name.

Table 3-11 Request parameter

Parameter	Mandatory	Type	Description
monitorGroup	No	String	Application name. For details, see **Querying the Application List**.

Example request

```
/v1/0/ats/applications/ams-metric:8080/instances?monitorGoup=11d5c9b83c1b2e04579fa5a34d191bb5
```

Response

Table 3-12 Response parameters

Parameter	Type	Description
errorCode	String	Error code. SVCSTG.ATS.2000: Query succeeded. SVCSTG.ATS.400101: Parameter verification failed. SVCSTG.ATS.200103: No instance information found.
errorMessage	String	Description of the error.
responseInfo	List (string)	Instance list of a specified service.

Example response

```
{
  "errorCode": "SVCSTG.ATS.2000",
  "errorMessage":null,
  "responseInfo": [ "d056db8ebf2350c118ea7ace383ac5dd"]
}
```

Status Code

- Success response

  Table 3-13 describes the status code.

Table 3-13 Status code

Status Code	Description
200	The request is successful.

3.4 Querying the Service Transaction List

Function

This API is used to query the transaction list of a specified service.

URI

GET /v1/{project_id}/ats/applications/{application}/transactions

Request

Table 3-14 Request headers

Parameter	Description	Mandatory	Example
clusterId	Cluster ID.	No	default
namespace	Namespace.	No	manage

Table 3-15 Path parameters

Parameter	Type	Description
project_id	String	Project ID.
application	String	Service name.

Table 3-16 Request parameter

Parameter	Mandatory	Type	Description
monitorGroup	Yes	String	Application name. For details, see **Querying the Application List**.

Example request

```
/v1/0/ats/applications/ams-metric:8080/transactions?monitorGroup=11d5c9b83c1b2e04579fa5a34d191bb5
```

Response

Table 3-17 Response parameters

Parameter	Type	Description
errorCode	String	Error code. SVCSTG.ATS.2000: Query succeeded. SVCSTG.ATS.400101: Parameter verification failed. SVCSTG.ATS.200103: No transaction information found.
errorMessage	String	Description of the error.

Parameter	Type	Description
responseInfo	List (string)	Transaction list of a specified service.

Example response

```
{
  "errorCode": "SVCSTG.ATS.2000",
  "errorMessage":null,
  "responseInfo": [
    "/amsalarm/v1/alarm/{project_id}"
  ]
}
```

Status Code

- Success response

  Table 3-18 describes the status code.

  Table 3-18 Status code

Status Code	Description
200	The request is successful.

3.5 Querying Tracing Data

Function

This API is used to query tracing data based on filter criteria.

URI

GET /v1/{project_id}/ats/traces

Request

Table 3-19 Request headers

Parameter	Description	Mandatory	Example
clusterId	Cluster ID.	No	default
namespace	Namespace.	No	manage

Table 3-20 Path parameter

Parameter	Type	Description
project_id	String	Project ID.

Table 3-21 Request parameters

Parameter	Mandatory	Type	Value Range	Description
startTime	Yes	Long	< endTime	Start time for querying tracing data (unit: ms).
endTime	Yes	Long	> startTime	End time for querying tracing data (unit: ms).
application	Yes	String	For details, see **Querying the Service List**.	Service name. The letters in the service name must be lowercase letters. Example: test-service.
monitorGroup	No	String	For details, see **Querying the Application List**.	Application name.
instance	No	String	For details, see **Querying the Service Instance List**.	Instance name. The letters in the instance name must be lowercase letters. Example: test-service-4195149926-0fvhn.
transaction	No	String	For details, see **Querying the Service Transaction List**.	Transaction name. Example: GET_/rest/healthz/*.
limit	No	Integer	The value must be an integer greater than 0 but less than or equal to 1000.	Maximum number of data records that can be returned each time. Default value: 20. Maximum value: 1000.

Parameter	Mandatory	Type	Value Range	Description
duration	No	Integer	The value must be an integer greater than or equal to 0.	Minimum call duration (unit: ms). Default value: 0.
status	No	Integer	**1**: Only the data of failed transactions is queried.	Transaction status. By default, all transaction data is queried. **1**: Only the data of failed transactions is queried.

Example request

```
/v1/0/ats/traces?
startTime=1506214200000&endTime=1506214428000&application=datamgmtservice&monitorGroup=apm&l
imit=1
```

Response

Table 3-22 Response parameters

Parameter	Type	Description
errorCode	String	Error code. SVCSTG.ATS.2000: Query succeeded. SVCSTG.ATS.400101: Parameter verification failed. SVCSTG.ATS.200103: No tracing data found.
errorMessage	String	Description of the error.
responseInfo	Result	Tracing query result.

Table 3-23 result parameters

Parameter	Type	Description
count	Integer	Number of traces that are queried.
traceChains	List<TraceChainBase>	Tracing data set.

Table 3-24 TraceChainBase parameters

Parameter	Type	Description
traceId	String	Trace ID, which is globally unique.
type	String	Service type.
status	Integer	Call response status.
duration	Long	Service call duration (unit: μs).
application	String	Service name.
instance	String	Instance name.
transaction	String	API/service name.
startTime	Long	Start time for calling a service (unit: μs).
endTime	Long	End time for calling a service (unit: μs).
address	String	IPv4 address of the client.

Example response

```
{
  "errorCode": "SVCSTG.ATS.2000",
  "errorMessage":null,
  "responseInfo": {
    "count": 1,
    "traceChains": [
      {
        "traceId": "000000004fa102d1",
        "type": "TOMCAT_METHOD",
        "status": 0,
        "duration": 10000,
        "application": "datamgmtservice",
        "instance": "datamgmtservice-4267750592-2ngmz",
        "transaction": "/rest/plat/sysmgr/v1/sysagent/alarm/report",
        "startTime": 1506214214095000,
        "endTime": 1506214214105000,
        "address": "192.168.0.1"
      }
    ]
  }
}
```

Status Code

- Success response

  Table 3-25 describes the status code.

Table 3-25 Status code

Status Code	Description
200	The request is successful.

3.6 Querying Tracing Details

Function

This API is used to query tracing details based on trace ID.

URI

GET /v1/{project_id}/ats/spans

Request

Table 3-26 Request headers

Parameter	Description	Mandatory	Example
clusterId	Cluster ID.	No	default
namespace	Namespace.	No	manage

Table 3-27 Path parameter

Parameter	Type	Description
project_id	String	Project ID.

Table 3-28 Request parameter

Parameter	Mandatory	Type	Value Range	Description
traceId	Yes	String	Trace ID obtained from the tracing data	Trace ID.

Example request

```
/v1/0/ats/spans?traceId=0000000027046b00
```

Response

Table 3-29 Response parameters

Parameter	Type	Description
errorCode	String	Error code. SVCSTG.ATS.2000: Query succeeded. SVCSTG.ATS.400101: Parameter verification failed. SVCSTG.ATS.200103: No tracing data found.
errorMessage	String	Description of the error.
responseInfo	List (string). For details, see **Table 3-30**.	Tracing query result.

Table 3-30 spans parameters

Parameter	Type	Description
traceId	String	Trace ID, which is globally unique.
name	String	Service name:Instance name:Transaction name.
id	String	Span ID.
parentId	String	Upper-level span ID.
timestamp	Long	Call start time (unit: μs).
duration	Long	Span call duration (unit: μs).
annotations	List (string). For details, see **Table 3-31**.	Service information about the client or server.
binaryAnnotations	List (string). For details, see **Table 3-32**.	Extended information.

Table 3-31 Annotation parameters

Parameter	Type	Description
timestamp	Long	Current system time when an event occurs (unit: μs).
endpoint	See **Table 3-33**.	(Optional) Service information about the client.
value	String	Invocation event type. Options: ● CS: The client sends a request. ● CR: The client receives a request. ● SR: The server receives a request. ● SS: The server sends a request.

Table 3-32 BinarryAnnotation parameters

Parameter	Type	Description
key	String	Name of the extended information.
endpoint	See **Table 3-33**.	(Optional) Service information about the client.
value	String	Value of the extended information.

Table 3-33 Endpoint parameters

Parameter	Type	Description
serviceName	String	(Optional) Service name of the client.
ipv4	String	(Optional) IP address of the client.
port	String	(Optional) Port of the client.

Example response

```
{
  "errorCode": "SVCSTG.ATS.2000",
  "errorMessage":null,
  "responseInfo": [
    "{\"traceId\":\"0000000027046b00\",\"id\":\"b42460f5cf86cab4\",\"name\":\"aos-apiserver:aos-
apiserver-1005774711-ll63p:/api/v1/namespaces/manage/pods\",\"timestamp\":
1506260836597000,\"duration\":67000,\"annotations\":[{\"timestamp\":1506260836597000,\"value\":\"cs\",
\"endpoint\":{\"serviceName\":\"aos-apiserver\",\"ipv4\":\"10.186.60.43\",\"port\":6443}},{\"timestamp\":
1506260836664000,\"value\":\"cr\",\"endpoint\":{\"serviceName\":\"aos-apiserver\",\"ipv4\":\"10.186.60.43\",
\"port\":6443}}],\"binaryAnnotations\":[{\"key\":\"append\",\"value\":\"GET\"},{\"key\":\"async\",\"value\":
\"0\"},{\"key\":\"goid\",\"value\":\"58\"},{\"key\":\"result\",\"value\":\"0\"},{\"key\":\"resultCode\",\"value\":
\"200\"},{\"key\":\"seqno\",\"value\":\"1506260836597048618\"},{\"key\":\"type\",\"value\":\"1\"}]}"
  ]
}
```

Status Code

- Success response

  Table 3-34 describes the status code.

  Table 3-34 Status code

Status Code	Description
200	The request is successful.

4 Permissions Policies and Supported Actions

4.1 Introduction

This chapter describes fine-grained permissions management for your APM. If your account does not need individual IAM users, then you may skip over this chapter.

By default, new IAM users do not have any permissions assigned. You need to add a user to one or more groups, and assign permissions policies or roles to these groups. The user then inherits permissions from the groups it is a member of. This process is called authorization. After authorization, the user can perform specified operations on APM.

You can grant users permissions by using roles**roles** and policies**policies**. Roles are a type of coarse-grained authorization mechanism that defines permissions related to user responsibilities. Policies define API-based permissions for operations on specific resources under certain conditions, allowing for more fine-grained, secure access control of cloud resources.

NOTE

Policy-based authorization is useful if you want to allow or deny the access to an API.

An account has all of the permissions required to call all APIs, but IAM users must have the required permissions specifically assigned. The permissions required for calling an API are determined by the actions supported by the API. Only users who have been granted permissions allowing the actions can call the API successfully. For example, if an IAM user queries metrics using an API, the user must have been granted permissions that allow the **apm:metric:get** action.

Supported Actions

IAM provides system-defined policies that can be directly used. 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 lists common concepts related to policies:

- Permissions: Defined by actions in custom policies.
- APIs: REST APIs that can be called by a user who has been granted specific permissions.
- Actions: Specific operations that are allowed or denied.
- Dependent actions: Actions on which a specific action depends to take effect. When assigning permissions for the action to a user, you also need to assign permissions for the dependent actions.
- IAM and enterprise projects: Type of projects for which an action will take effect. Policies that contain actions for both IAM and enterprise projects can be used and take effect for both IAM and Enterprise Management. Policies that only contain actions for IAM projects can be used and only take effect for IAM.

APM supports the following actions that can be defined in custom policies:

- **Actions**: includes the actions supported by APM APIs, such as the APIs for querying the application list, service list, service instance list, service transaction list, tracing data, and tracing details.

4.2 Actions

NOTE

√: supported; x: not supported

Table 4-1 API actions

Permissions	API	Action	IAM Project	Enterprise Project
Querying the application list	GET /v1/{project_id}/atps/monitorgroups	apm:inventory:get	√	√
Querying the service list	GET /v1/{project_id}/ats/applications	apm:ats:get	√	√
Querying the service instance list	GET /v1/{project_id}/ats/applications/{application}/instances	apm:ats:get	√	√

Permissions	API	Action	IAM Project	Enterprise Project
Querying the service transaction list	GET /v1/{project_id}/ats/applications/{application}/transactions	apm:ats:get	√	√
Querying tracing data	GET /v1/{project_id}/ats/traces	apm:ats:get	√	√
Querying tracing details	GET /v1/{project_id}/ats/spans	apm:ats:get	√	√

5 Appendix

5.1 Status Code

Table 5-1 lists the status codes.

Table 5-1 Status codes

Status Code	Message	Description
100	Continue	The server has received the initial part of the request and the client should continue to send the remaining part. This code is issued on a provisional basis while request processing continues. It alerts the client to wait for a final response.
101	Switching Protocols	The protocol should be switched. The target protocol must be more advanced than the source protocol. For example, the current HTTP protocol is switched to a later version of HTTP.
200	OK	The server has successfully processed the request.

Status Code	Message	Description
201	Created	The request has been fulfilled, resulting in the creation of a new resource.
202	Accepted	The request has been accepted for processing, but the processing has not been completed.
203	Non-Authoritative Information	The server successfully processed the request, but is returning information that may be from another source.
204	No Content	The server has successfully processed the request, but does not return any content. The status code is returned in response to an HTTP OPTIONS request.
205	Reset Content	The server has successfully processed the request, but does not return any content. This response requires that the requester reset the content.
206	Partial Content	The server has successfully processed a part of the GET request.
300	Multiple Choices	There are multiple options for the requested resource. For example, this code could be used to present a list of resource characteristics and addresses from which the client such as a browser may choose.
301	Moved Permanently	This and all future requests should be permanently directed to the given URI indicated in this response.

Status Code	Message	Description
302	Found	The requested resource was temporarily moved.
303	See Other	The response to the request can be found under another URI using a GET or POST method.
304	Not Modified	The requested resource has not been modified. In such case, there is no need to retransmit the resource since the client still has a previously-downloaded copy.
305	Use Proxy	The requested resource is available only through a proxy.
306	Unused	This HTTP status code is no longer used.
400	Bad Request	The request is invalid. The client should modify the request instead of re-initiating it.
401	Unauthorized	The authorization information provided by the client is incorrect or invalid.
402	Payment Required	This status code is reserved for future use.
403	Forbidden	The server has received the request and understood it, but the server is refusing to respond to it. The client should modify the request instead of re-initiating it.
404	Not Found	The requested resource could not be found. The client should modify the request instead of re-initiating it.

Status Code	Message	Description
405	Method Not Allowed	A request method is not supported for the requested resource. The client should modify the request instead of re-initiating it.
406	Not Acceptable	The server could not fulfill the request according to the content characteristics of the request.
407	Proxy Authentication Required	This code is similar to 401, but indicates that the client must first authenticate itself with the proxy.
408	Request Time-out	The server timed out waiting for the request. The client may re-initiate the request without modifications at any later time.
409	Conflict	The request could not be processed due to a conflict in the request, such as an edit conflict between multiple simultaneous updates or the resource that the client attempts to create exists.
410	Gone	The requested resource has been deleted permanently and will not be available again.
411	Length Required	The server refused to process the request because the request does not specify the length of its content.
412	Precondition Failed	The server does not meet one of the preconditions that the requester puts on the request.

Status Code	Message	Description
413	Request Entity Too Large	The server refuses to process a request because the request is too large. The server may disable the connection to prevent the client from sending requests consecutively. If the server temporarily cannot process the request, the response will contain a Retry-After header field.
414	Request-URI Too Large	The URI provided was too long for the server to process.
415	Unsupported Media Type	The server does not support the media type in the request.
416	Requested range not satisfiable	The requested range is invalid.
417	Expectation Failed	The server fails to meet the requirements of the Expect request-header field.
422	Unprocessable Entity	The request was well-formed but was unable to be followed due to semantic errors.
429	Too Many Requests	The client sends excessive requests to the server within a given time (exceeding the limit on the access frequency of the client), or the server receives excessive requests within a given time (beyond its processing capability). In this case, the client should repeat requests after the time specified in the Retry-After header of the response expires.

Status Code	Message	Description
500	Internal Server Error	The server is able to receive the request but unable to understand the request.
501	Not Implemented	The server does not support the requested function.
502	Bad Gateway	The server was acting as a gateway or proxy and received an invalid request from a remote server.
503	Service Unavailable	The requested service is invalid. The client should modify the request instead of re-initiating it.
504	Server Timeout	The request cannot be fulfilled within a given time. This status code is returned to the client only when the timeout parameter is specified in the request.
505	HTTP Version not supported	The server does not support the HTTP protocol version used in the request.

5.2 Error Code

If an error occurs during API calling, no result will be returned. You can locate the cause based on the error code of each API. If an error occurs during API calling, HTTP status code 4xx or 5xx is returned. The response body contains the specific error code and information. If you fail to locate the cause of an error, contact customer service and provide the error code for fast troubleshooting.

Format of an Error Response Body

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

```
{
    "errorCode": "SVCSTG_AMS_4000001",
    "errorMessage": "Request param invalid"
}
```

In the preceding information, **errorCode** is an error code, and **errorMessage** describes the error.

Error Code Description

If an error code starting with **APIGW** is returned after you call an API, rectify the fault according to **Error Codes**.

Table 5-2 Error codes

Status Code	Error Code	Message	Description	Solution
200	SVCSTG.ATPS.2000	null	-	-
200	SVCSTG.ATS.2000	null	-	-
400	SVCSTG.ATS.400101	projectId in url is invalid.	Invalid projectId in the URL.	Check whether the parameter meets requirements.
400	SVCSTG.ATS.200103	query trace result is empty.	The trace query result is empty.	Check whether the parameter meets requirements.

5.3 Obtaining a Project ID

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 from the console, perform the following operations:

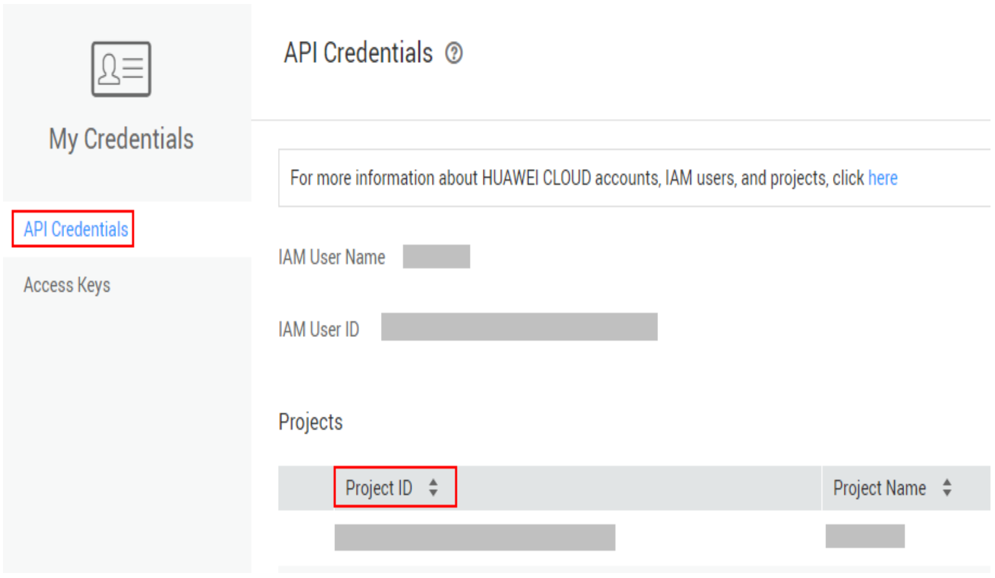
Step 1 Sign up and log in to the management console.

Step 2 Hover the mouse pointer over the username and choose **Basic Information** from the drop-down list.

Step 3 On the displayed **Basic Information** page, click **Manage**.

On the displayed **API Credentials** page, view project IDs in the project list.

Figure 5-1 Viewing project IDs



If a project contains multiple sub-projects, click the plus (+) sign to view sub-project IDs.

----End

Obtaining a Project ID by Calling an API

You can also call the API for **querying project information** to obtain a project ID.

The API is **GET https://*{Endpoint}*/v3/projects/**, where *{Endpoint}* indicates the Identity and Access Management (IAM) endpoint. For details, see **Regions and Endpoints**.

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

```
{
    "projects": [
        {
            "domain_id": "65382450e8f64ac0870cd180d14e684b",
            "is_domain": false,
            "parent_id": "65382450e8f64ac0870cd180d14e684b",
            "name": "ap-southeast-1",
            "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"
    }
}
```

5.4 Obtaining an Account ID

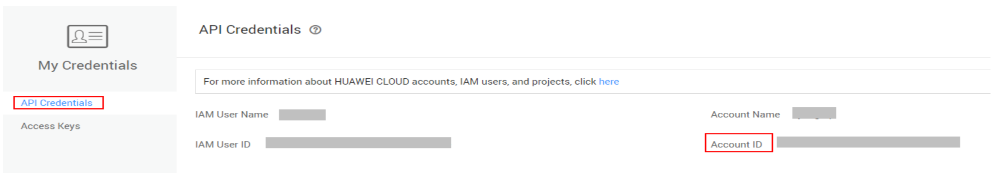
An account ID is required for some URLs when an API is called. To obtain an account ID, perform the following operations:

Step 1 Sign up and log in to the management console.

Step 2 Hover the mouse pointer over the username and choose **My Credentials** from the drop-down list.

On the **API Credentials** page, view **Account ID**.

Figure 5-2 Obtaining an account ID



API Credentials

For more information about HUAWEI CLOUD accounts, IAM users, and projects, click here

IAM User Name [redacted]

Account Name [redacted]

IAM User ID [redacted]

Account ID [redacted]

----End

5.5 Common Request Headers

Table 5-3 Common request headers

Header	Description	Mandatory	Example
X-Auth-Token	User token.	Mandatory for token authentication.	-
Content-Type	Type of content in a request. The value of this field is **application/ json;charset=utf8**.	Mandatory	application/ json;charset=utf8
x-sdk-date	Time to send a request. The format is **YYYYMMDD'T'HHMMSS'Z**. GMT time is used.	Mandatory for AK/SK authentication.	20160629T101459Z
Authorization	Authentication information, which is the result of request signing.	Mandatory for AK/SK authentication.	-

Header	Description	Mandatory	Example
Host	Requested server information, which is obtained from the URL in an API request. The value is in the format of *hostname:port*. If no port is specified, the default port will be used. For HTTPS, port 443 is used by default.	Mandatory for AK/SK authentication.	-

5.6 Common Response Headers

A response usually contains the following headers:

Table 5-4 Common response headers

Header	Description	Example
Date	(Standard HTTP header) Time when a message is sent. The time format is defined by RFC822.	Mon, 12 Nov 2007 15:55:01 GMT
Server	(Standard HTTP header) Software information used by the server to process a request.	Apache
Content-Length	(Standard HTTP header) Size of the response body, in decimal number of bytes.	xxx
Content-Type	(Standard HTTP header) Media type of the response body sent to the recipient.	application/json

6 Change History

Date	Description
2020-08-21	Supported API actions for Enterprise Project Management Service (EPS). For details, see **Actions**.
2018-05-30	This issue is the first official release.