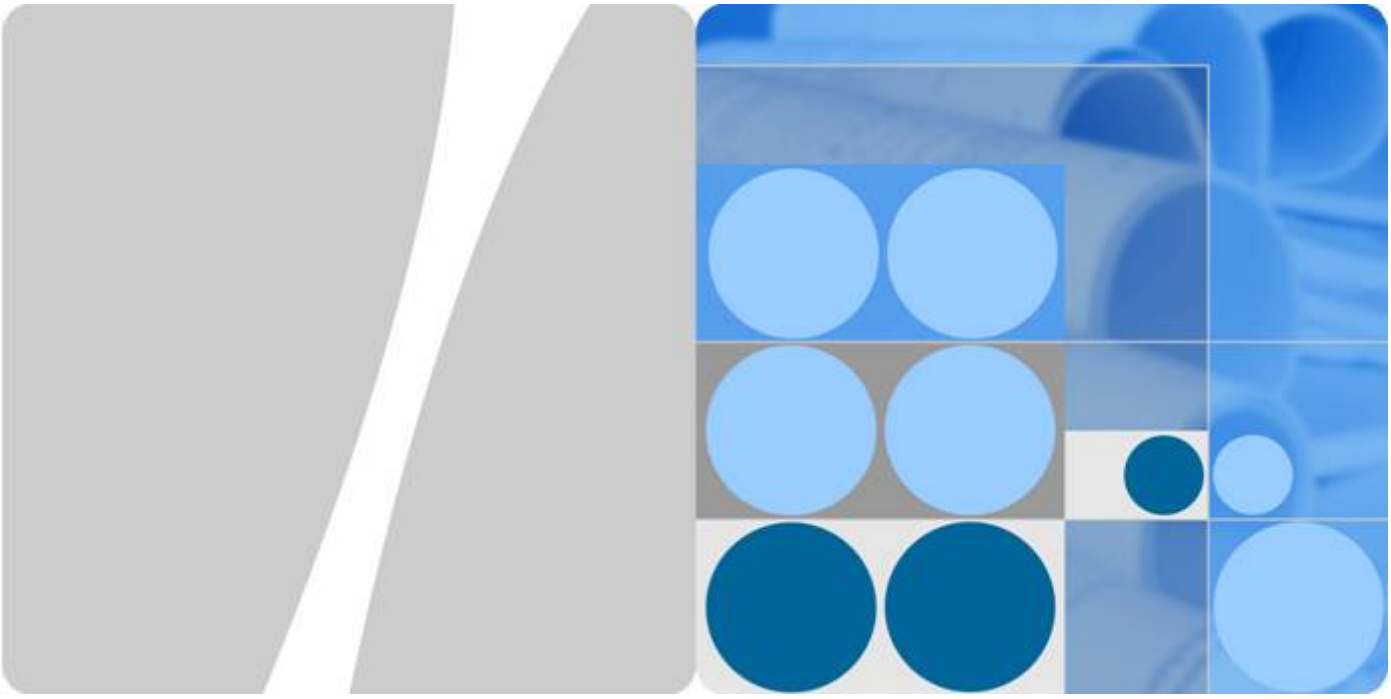


Tag Management Service

API Reference

Date 2025-03-31

Contents

1 Before You Start

Welcome to Tag Management Service (TMS). Tags are useful for identifying cloud resources, especially when you have a good many resources of the same type. You can classify resources by usage, ownership, or environment. TMS is a visualized service that allows you to efficiently and centrally manage tags and categorize cloud resources across regions and services.

This document describes how to use application programming interfaces (APIs) to perform operations on tags, such as creating or deleting predefined tags, and querying or modify predefined tags. For details about all supported operations, see **API Overview**.

Before using TMS APIs, ensure that you are familiar with TMS concepts. For details, see section "Service Overview" in the *Tag Management Service User Guide*.

TMS supports Representational State Transfer (REST) APIs, allowing you to call APIs using HTTPS. For details about API calling, see **API Calling**.

Endpoints

An endpoint is a **request address** for calling an API. Endpoints vary depending on services and regions. Obtain the regions and endpoints from the enterprise administrator.

Basic Concepts

- Domain

  A domain has full access permissions for all of its cloud services and resources. It can be used to reset user passwords and grant user permissions. The domain should not be used directly to perform routine management. For security purposes, create Identity and Access Management (IAM) users and grant them permissions for routine management.

- User

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

  API authentication requires information such as the domain name, username, and password.

- Region

A region is a geographic area in which cloud resources are deployed. Availability zones (AZs) in the same region can communicate with each other over an intranet, while AZs in different regions are isolated from each other. Deploying cloud resources in different regions can better suit certain user requirements or comply with local laws or regulations.

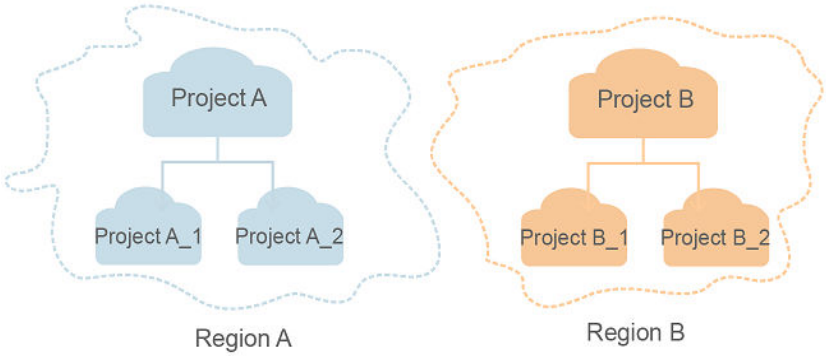
- AZ

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

- Project

  A project corresponds to a region. Default projects are defined to group and physically isolate resources (including computing, storage, and network resources) across regions. Users can be granted permissions in a default project to access all resources under their domains in the region associated with the project. If you need more refined access control, create subprojects under a default project and create resources in subprojects. Then you can assign users the permissions required to access only the resources in the specific subprojects.

Figure 1-1 Project isolation model



2 API Overview

Table 2-1 TMS APIs

API	Description
Querying the API Version	API for querying TMS API versions
Querying Details About an API Version	API for querying details about a specified TMS API version
Creating Predefined Tags	API for creating or deleting predefined tags. You can use predefined tags to tag resources.
Deleting Predefined Tags	API for deleting predefined tags. You can delete predefined tags created.
Query a Predefined Tag List	API for querying predefined tags of a specified user
Modifying Predefined Tags	API for modifying a predefined tag
Querying Services Supported by TMS	API for querying cloud services supported by TMS

3 API Calling

3.1 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.

Table 3-1 URI parameter description

Parameter	Description
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.

Table 3-2 HTTP methods

Method	Description
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.

Table 3-3 Common request header fields

Parameter	Description	Mandatory	Example Value
Host	Specifies the server domain name and port number of the resources being requested. The value can be obtained from the URL of the service API. The value is in the format of *Hostname:Port number*. If the port number is not specified, the default port is used. The default port number for **https** is **443**.	No This field is mandatory for AK/SK authentication.	code.test.com or code.test.com:443
Content-Type	Specifies the type (or format) of the message body. The default value **application/json** is recommended. Other values of this field will be provided for specific APIs if any.	Yes	application/json
Content-Length	Specifies the length of the request body. The unit is byte.	No	3495
X-Project-Id	Specifies the project ID. Obtain the project ID by following the instructions in **Obtaining a Project ID**.	No	e9993fc787d94b6c886cbaa340f9c0f4
X-Auth-Token	Specifies the user token. It is a response to the API for obtaining a user token (This is the only API that does not require authentication). After the request is processed, the value of **X-Subject-Token** in the response header is the token value.	No This field is mandatory for token authentication.	The following is part of an example token: MIIPAgYJKoZIhvcNAQcCo...ggg1BBIINPXsidG9rZ

> **NOTE**
>
> In addition to supporting authentication using tokens, APIs support authentication using AK/SK, which uses SDKs to sign a request. During the signature, the **Authorization** (signature authentication) and **X-Sdk-Date** (time when a request is sent) headers are automatically added in the request.
>
> For more details, see "Authentication Using AK/SK" in **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:

```
POST https://{{endpoint}}/v3/auth/tokens
Content-Type: application/json
```

(Optional) Request Body

This part is optional. The body of a request is often sent in a structured format (for example, JSON or XML) 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. Replace *username*, *domainname*, *$ADMIN_PASS* (login password), and *xxxxxxxxxxxxxxxxx* (project name) with the actual values. Obtain a project name from the administrator.

> **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": "$ADMIN_PASS",     //You are advised to store it in ciphertext in the configuration file or an environment variable and decrypt it when needed to ensure security.
                    "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.

3.2 Authentication

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

- Token authentication: Requests are authenticated using tokens.
- AK/SK authentication: Requests are encrypted using AK/SK pairs. AK/SK authentication is recommended because it is more secure than token authentication.

Token Authentication

NOTE

The validity period of a token is 24 hours. When using a token for authentication, cache it to prevent frequently calling the IAM API used to obtain a user token.

A token specifies temporary permissions in a computer system. During API authentication using a token, the token is added to requests to get permissions for calling the API. You can obtain a token by calling the Obtaining a User Token API.

A cloud service can be deployed as either a project-level service or global service.

- For a project-level service, you need to obtain a project-level token. When you call the API, set **auth.scope** in the request body to **project**.
- For a global service, you need to obtain a global token. When you call the API, set **auth.scope** in the request body to **domain**.

TMS is a global service. When you call the API, set **auth.scope** in the request body to **domain**. For details about how to obtain the user token, see Obtaining a User Token.

```
{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "username",   // IAM user name
                    "password": "********",  // IAM user password
                    "domain": {
                        "name": "domainname"  // Name of the account to which the IAM user belongs
                    }
                }
            }
        },
        "scope": {
            "domain": {
                "name": "xxxxxxxx"    // Tenant name
            }
        }
```

```
    }
}
```

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

```
POST https://{{endpoint}}/v3/auth/projects
Content-Type: application/json
X-Auth-Token: ABCDEFJ....
```

3.3 Response

Status Code

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

A status code is a group of digits, ranging from 1xx to 5xx. It indicates the status of a request. For more information, see **Status Codes**.

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

Response Header

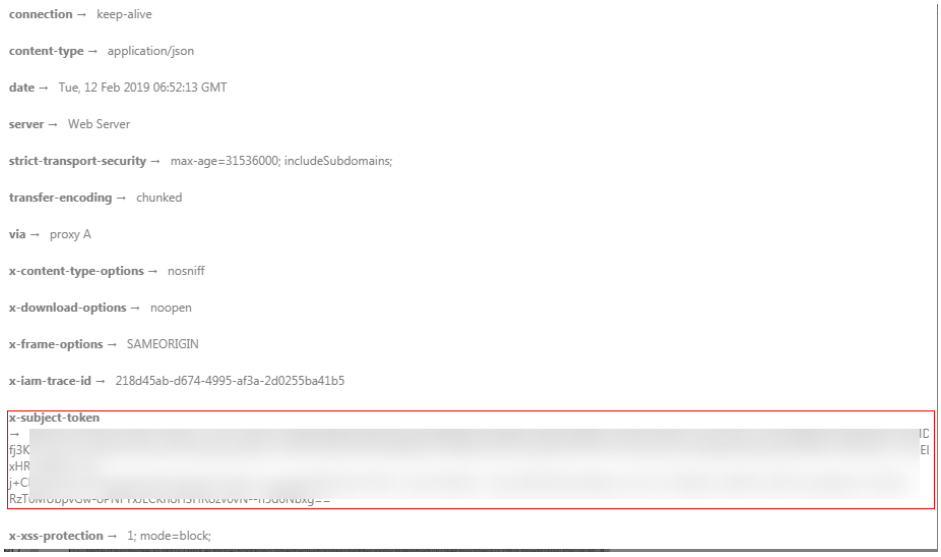
Similar to a request, a response also has a header, for example, **Content-Type**.

Figure 3-1 shows the response header fields for the API used to obtain a user token. The **X-Subject-Token** header field is the desired user token. This token can then be used to authenticate the calling of other APIs.

NOTE

For security purposes, you are advised to set the token in ciphertext in configuration files or environment variables and decrypt it when using it.

Figure 3-1 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 a structured format (for example, JSON or XML) 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 used to obtain a user token.

```
{
    "token": {
        "expires_at": "2019-02-13T06:52:13.855000Z",
        "methods": [
            "password"
        ],
        "catalog": [
            {
                "endpoints": [
                    {
                        "region_id": "az-01",
.....
```

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 request message format is invalid.",
    "error_code": "IMG.0001"
}
```

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

4 API Description

4.1 API Version Querying

4.1.1 Querying API Versions

Function

Querying API versions

Calling Method

For details, see **Calling APIs**.

URI

GET /

Request Parameters

None

Response Parameters

Status code: 200

Table 4-1 Response body parameters

Parameter	Type	Description
versions	Array of **VersionDetail** objects	List of versions

Table 4-2 VersionDetail

Parameter	Type	Description
id	String	Specifies the version ID, for example, v1.0.
links	Array of **Link** objects	Specifies the API URL.
version	String	If there are minor versions, the earliest minor version is returned. If there are no minor versions, null is returned.
status	String	Specifies the version status. Possible values are as follows: **CURRENT**: widely used version **SUPPORTED**: earlier version which is still supported **DEPRECATED**: deprecated version which may be deleted later
updated	String	Specifies the version release time, which is a UTC time. For example, the release time of v1.0 is 2016-12-09T00:00:00Z.
min_version	String	If there are minor versions, the earliest minor version is returned. If there are no minor versions, null is returned.

Table 4-3 Link

Parameter	Type	Description
href	String	The API URL.
rel	String	self

Status code: 400

Table 4-4 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-5 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 404

Table 4-6 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-7 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 405

Table 4-8 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-9 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 406

Table 4-10 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-11 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 409

Table 4-12 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-13 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 410

Table 4-14 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-15 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 412

Table 4-16 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-17 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 429

Table 4-18 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-19 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 500

Table 4-20 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-21 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 501

Table 4-22 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-23 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 503

Table 4-24 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-25 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Example Requests

Querying API versions

```
GET https://{Endpoint}/
```

Example Responses

Status code: 200

OK

```
{
  "versions" : [ {
    "id" : "v1.0",
    "links" : [ {
      "rel" : "self",
      "href" : "https://{Endpoint}/v1.0"
    } ],
    "version" : "",
    "status" : "CURRENT",
    "updated" : "2016-12-09T00:00:00Z",
    "min_version" : ""
  } ]
}
```

Status Codes

Status Code	Description
200	OK
400	Bad Request
404	Not Found
405	Method Not Allowed
406	Not Acceptable
409	Conflict
410	Gone
412	Precondition Failed
429	Too Many Requests
500	Internal Server Error

Status Code	Description
501	Not Implemented
503	Service Unavailable

Error Codes

See **Error Codes**.

4.1.2 Querying Details About an API Version

Function

This API is used to query details about a specified TMS API version.

Calling Method

For details, see **Calling APIs**.

URI

GET /{api_version}

Table 4-26 Path Parameters

Parameter	Mandatory	Type	Description
api_version	Yes	String	Specifies the API version.

Request Parameters

Table 4-27 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	Specifies the user token. TMS is a global service. When calling the Identity and Access Management (IAM) API to obtain a user token, set the **scope** field to **domain**. The value of **X-Subject-Token** in the response header is the user token.

Response Parameters

Status code: 200

Table 4-28 Response body parameters

Parameter	Type	Description
version	**VersionDetail** object	Specifies the version details.

Table 4-29 VersionDetail

Parameter	Type	Description
id	String	Specifies the version ID, for example, v1.0.
links	Array of **Link** objects	Specifies the API URL.
version	String	If there are minor versions, the earliest minor version is returned. If there are no minor versions, null is returned.
status	String	Specifies the version status. Possible values are as follows: **CURRENT**: widely used version **SUPPORTED**: earlier version which is still supported **DEPRECATED**: deprecated version which may be deleted later
updated	String	Specifies the version release time, which is a UTC time. For example, the release time of v1.0 is 2016-12-09T00:00:00Z.
min_version	String	If there are minor versions, the earliest minor version is returned. If there are no minor versions, null is returned.

Table 4-30 Link

Parameter	Type	Description
href	String	The API URL.
rel	String	self

Status code: 400

Table 4-31 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-32 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 404

Table 4-33 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-34 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 405

Table 4-35 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-36 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 406

Table 4-37 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-38 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 408

Table 4-39 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-40 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 409

Table 4-41 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-42 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 410

Table 4-43 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-44 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 412

Table 4-45 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-46 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 429

Table 4-47 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-48 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 500

Table 4-49 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-50 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 501

Table 4-51 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-52 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 503

Table 4-53 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-54 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Example Requests

Querying details about a TMS API version

```
GET https://{Endpoint}/v1.0
```

Example Responses

Status code: 200

OK

```
{
  "version" : {
    "id" : "v1.0",
    "links" : [ {
      "rel" : "self",
      "href" : "https://{Endpoint}/v1.0"
```

```
    } ],
    "version" : "",
    "status" : "CURRENT",
    "updated" : "2016-12-09T00:00:00Z",
    "min_version" : ""
  }
}
```

Status Codes

Status Code	Description
200	OK
400	Bad Request
404	Not Found
405	Method Not Allowed
406	Not Acceptable
408	Request Timeout
409	Conflict
410	Gone
412	Precondition Failed
429	Too Many Requests
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

Error Codes

See **Error Codes**.

4.2 Predefined Tags

4.2.1 Creating Predefined Tags

Function

This API is used to create predefined tags. This API supports idempotency and batch data processing.

Calling Method

For details, see **Calling APIs**.

URI

POST /v1.0/predefine_tags/action

Request Parameters

Table 4-55 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	Specifies the user token. TMS is a global service. When calling the Identity and Access Management (IAM) API to obtain a user token, set the **scope** field to **domain**. The value of **X-Subject-Token** in the response header is the user token.

Table 4-56 Request body parameters

Parameter	Mandatory	Type	Description
action	Yes	String	Specifies the operation (case sensitive). The value is **create**.
tags	Yes	Array of **PredefineTagRequest** objects	Tag list.

Table 4-57 PredefineTagRequest

Parameter	Mandatory	Type	Description
key	Yes	String	Specifies the tag key. The value can contain up to 36 characters including letters, digits, hyphens (-), and underscores (_).
value	Yes	String	Specifies the tag value. The value can contain up to 43 characters including letters, digits, periods (.), hyphens (-), and underscores (_). It can be an empty string.

Response Parameters

Status code: 400

Table 4-58 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-59 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 401

Table 4-60 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-61 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 403

Table 4-62 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-63 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 404

Table 4-64 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-65 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 405

Table 4-66 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-67 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 406

Table 4-68 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-69 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 409

Table 4-70 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-71 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 410

Table 4-72 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-73 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 412

Table 4-74 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-75 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 429

Table 4-76 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-77 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 500

Table 4-78 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-79 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 501

Table 4-80 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-81 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 503

Table 4-82 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-83 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Example Requests

Creating predefined tags

```
POST https://{Endpoint}/v1.0/predefine_tags/action

{
  "action" : "create",
  "tags" : [ {
    "key" : "ENV1",
    "value" : "DEV1"
  }, {
    "key" : "ENV2",
    "value" : "DEV2"
  } ]
}
```

Example Responses

None

Status Codes

Status Code	Description
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
409	Conflict
410	Gone
412	Precondition Failed
429	Too Many Requests
500	Internal Server Error

Status Code	Description
501	Not Implemented
503	Service Unavailable

Error Codes

See **Error Codes**.

4.2.2 Deleting Predefined Tags

Function

This API is used to delete predefined tags. This API supports idempotency and batch data processing.

Calling Method

For details, see **Calling APIs**.

URI

POST /v1.0/predefine_tags/action

Request Parameters

Table 4-84 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token. Note: TMS is a global service. Therefore, when calling the IAM service to obtain user tokens, set the scope field to domain. The value of X-Subject-Token in the response header is the user token.

Table 4-85 Request body parameters

Parameter	Mandatory	Type	Description
action	Yes	String	Specifies the operation (case sensitive). The value is **delete**.

Parameter	Mandatory	Type	Description
tags	Yes	Array of **PredefineTagRequest** objects	Specifies a tag list. Up to 50 tags can be deleted at the same time.

Table 4-86 PredefineTagRequest

Parameter	Mandatory	Type	Description
key	Yes	String	Specifies the tag key. The value can contain up to 36 characters including letters, digits, hyphens (-), and underscores (_).
value	Yes	String	Specifies the tag value. The value can contain up to 43 characters including letters, digits, periods (.), hyphens (-), and underscores (_). It can be an empty string.

Response Parameters

Status code: 400

Table 4-87 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-88 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 401

Table 4-89 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-90 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 403

Table 4-91 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-92 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 404

Table 4-93 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-94 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 405

Table 4-95 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-96 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 406

Table 4-97 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-98 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 409

Table 4-99 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-100 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 410

Table 4-101 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-102 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 412

Table 4-103 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-104 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 429

Table 4-105 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-106 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 500

Table 4-107 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-108 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 501

Table 4-109 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-110 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 503

Table 4-111 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-112 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Example Requests

Deleting predefined tags

```
POST https://{Endpoint}/v1.0/predefine_tags/action

{
  "action" : "delete",
  "tags" : [ {
    "key" : "ENV1",
    "value" : "DEV1"
  }, {
    "key" : "ENV2",
    "value" : "DEV2"
  } ]
}
```

Example Responses

None

Status Codes

Status Code	Description
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
409	Conflict
410	Gone
412	Precondition Failed
429	Too Many Requests
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

Error Codes

See **Error Codes**.

4.2.3 Querying Predefined Tags

Function

This API is used to query predefined tags.

Calling Method

For details, see **Calling APIs**.

URI

GET /v1.0/predefine_tags

Table 4-113 Query Parameters

Parameter	Mandatory	Type	Description
key	No	String	Specifies the tag key. Fuzzy search is supported. **Key** is case insensitive. If the key contains non-URL-safe characters, it must be URL encoded.
value	No	String	Specifies the tag value. Fuzzy search is supported. **Value** is case insensitive. If the value contains non-URL-safe characters, it must be URL encoded.
limit	No	Integer	Specifies the number of records to be queried, which is **10** by default. The maximum value is **1000** and the minimum value is **1**. If the value is **0**, the number of records to be queried is not limited.
marker	No	String	Specifies the paging location marker (index position). The query starts from the next piece of data of the index specified by **marker**.Note:You do not need to specify this parameter when you query the data on the first page. When you query the data on subsequent pages, set this parameter to the **marker** value returned in the response body for the previous query. If the returned **tags** is empty, the last page is queried.

Parameter	Mandatory	Type	Description
order_field	No	String	Specifies the sorting field. The value can be **update_time**, **key**, or **value**. The value is case sensitive. You can sort tags based on the value of **order_method**. If this value is not specified, the default value is **update_time**. For example: If **order_field** is set to **update_time**, values of **key** and **value** are sorted in ascending order. If **order_field** is set to **key**, values of **update_time** are sorted in descending order and **value** in ascending order. If **order_field** is set to **value**, values of **update_time** are sorted in descending order and **value** in ascending order. If **order_field** is not specified, the default value **update_time** is used, and values of **key** and **value** are sorted in ascending order.
order_method	No	String	Specifies the sorting method of **order_field**. The value can be **asc** or **desc**. The value is case sensitive. If this parameter is not specified, the default value is **desc**.

Request Parameters

Table 4-114 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	Specifies the user token. TMS is a global service. When calling the Identity and Access Management (IAM) API to obtain a user token, set the **scope** field to **domain**. The value of **X-Subject-Token** in the response header is the user token.

Response Parameters

Status code: 200

Table 4-115 Response body parameters

Parameter	Type	Description
marker	String	Specifies the paging location marker (index position).
total_count	Integer	Total number of queried tags.
tags	Array of **PredefineTag** objects	List of queried tags.

Table 4-116 PredefineTag

Parameter	Type	Description
key	String	Specifies the tag key. The value can contain up to 36 characters including letters, digits, hyphens (-), and underscores (_).
value	String	Specifies the tag value. The value can contain up to 43 characters including letters, digits, periods (.), hyphens (-), and underscores (_). It can be an empty string.
update_time	String	Update time, which must be the UTC time. 2016-12-09T00:00:00Z

Status code: 400

Table 4-117 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-118 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 401

Table 4-119 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-120 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 403

Table 4-121 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-122 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 404

Table 4-123 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-124 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 405

Table 4-125 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-126 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 406

Table 4-127 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-128 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 409

Table 4-129 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-130 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 410

Table 4-131 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-132 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 412

Table 4-133 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-134 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 429

Table 4-135 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-136 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 500

Table 4-137 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-138 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 501

Table 4-139 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-140 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 503

Table 4-141 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-142 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Example Requests

Querying predefined tags

```
GET https://{Endpoint}/v1.0/predefine_tags?
key=ENV&value=DEV&limit=10&marker=9&order_field=key&order_method=asc
```

Example Responses

Status code: 200

OK

```
{
  "marker" : "12",
  "total_count" : 13,
  "tags" : [ {
    "key" : "ENV1",
    "value" : "DEV1",
    "update_time" : "2017-04-12T14:22:34Z"
  }, {
    "key" : "ENV2",
    "value" : "DEV2",
    "update_time" : "2017-04-12T14:22:34Z"
  } ]
}
```

Status Codes

Status Code	Description
200	OK
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
409	Conflict
410	Gone
412	Precondition Failed

Status Code	Description
429	Too Many Requests
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

Error Codes

See **Error Codes**.

4.2.4 Modifying a Predefined Tag

Function

Modify predefined tags.

Calling Method

For details, see **Calling APIs**.

URI

PUT /v1.0/predefine_tags

Request Parameters

Table 4-143 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	Specifies the user token. TMS is a global service. When calling the Identity and Access Management (IAM) API to obtain a user token, set the **scope** field to **domain**. The value of **X-Subject-Token** in the response header is the user token.

Table 4-144 Request body parameters

Parameter	Mandatory	Type	Description
new_tag	Yes	**PredefineTagRequest** object	Specifies the modified tag.
old_tag	Yes	**PredefineTagRequest** object	Specifies the tag before modification.

Table 4-145 PredefineTagRequest

Parameter	Mandatory	Type	Description
key	Yes	String	Specifies the tag key. The value can contain up to 36 characters including letters, digits, hyphens (-), and underscores (_).
value	Yes	String	Specifies the tag value. The value can contain up to 43 characters including letters, digits, periods (.), hyphens (-), and underscores (_). It can be an empty string.

Response Parameters

Status code: 400

Table 4-146 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-147 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 401

Table 4-148 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-149 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 403

Table 4-150 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-151 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 404

Table 4-152 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-153 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 405

Table 4-154 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-155 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 406

Table 4-156 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-157 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 409

Table 4-158 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-159 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 410

Table 4-160 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-161 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 412

Table 4-162 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-163 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 429

Table 4-164 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-165 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 500

Table 4-166 Response body parameters

Parameter	Type	Description
error	**RespErrorMessage** object	Response error information.

Table 4-167 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 501

Table 4-168 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-169 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Status code: 503

Table 4-170 Response body parameters

Parameter	Type	Description
error	RespErrorMessage object	Response error information.

Table 4-171 RespErrorMessage

Parameter	Type	Description
error_code	String	Request error code
error_msg	String	Error message

Example Requests

Modifying a predefined tag

```
PUT https://{Endpoint}/v1.0/predefined_tags

{
  "new_tag" : {
    "key" : "ENV1",
    "value" : "DEV1"
  },
  "old_tag" : {
    "key" : "ENV2",
    "value" : "DEV2"
  }
}
```

Example Responses

None

Status Codes

Status Code	Description
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
409	Conflict
410	Gone
412	Precondition Failed
429	Too Many Requests
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

Error Codes

See **Error Codes**.

5 Permissions Policies and Supported Actions

5.1 Permissions Policies and Supported Actions

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

By default, new IAM users do not have permissions assigned. You need to add a user to one or more groups, and attach permissions policies or roles to these groups. Users inherit permissions from the groups to which they are added and can perform specified operations on cloud services based on the permissions.

You can grant users permissions by using roles and 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.

Each account has all the permissions required to call all APIs, but IAM users must be assigned the required permissions. 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 wants to query predefined tags using an API, the user must have been granted permissions that allow the **tms:predefineTags:list** action.

Supported Actions

Operations supported by a fine-grained policy are specific to APIs. The following are common concepts related to policies:

- Permissions: Statements in a policy that allow or deny certain operations.
- APIs: REST APIs that can be called by a user who has been granted specific permissions

- Actions: Specific operations that are allowed or denied.
- Dependencies: actions which a specific action depends on. When allowing an action for a user, you also need to allow any existing action dependencies for that user.
- IAM or 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 applied for both IAM and Enterprise Management. Policies that contain actions only for IAM projects can be used and applied to IAM only. Administrators can check whether an action supports IAM projects or enterprise projects in the action list.

5.2 TMS API Actions

Table 5-1 API actions

Permission	API	Action	IAM Project	Enterprise Project
Querying predefined tags	GET /v1.0/predefine_tags	tms:predefineTags:list	Supported	Not supported
Creating predefined tags	POST /v1.0/predefine_tags/action	tms:predefineTags:create	Supported	Not supported
Deleting predefined tags	POST /v1.0/predefine_tags/action	tms:predefineTags:delete	Supported	Not supported
Modifying a predefined tag	PUT /v1.0/predefine_tags	tms:predefineTags:update	Supported	Not supported

A Appendix

A.1 Status Codes

- Normal

Returned Value	Description
200 OK	The results of GET and PUT operations are returned as expected.
201 Created	The results of the POST operation are returned as expected.
202 Accepted	The request has been accepted for processing.
204 No Content	Normal response code

- Abnormal

Returned Value	Description
400 Bad Request	The server failed to process the request.
401 Unauthorized	You must enter a username and password to access the requested page.
403 Forbidden	Access to the requested page is denied.
404 Not Found	The server cannot find the requested page.
405 Method Not Allowed	You are not allowed to use the method specified in the request.
406 Not Acceptable	The response generated by the server cannot be accepted by the client.

Returned Value	Description
407 Proxy Authentication Required	You must use the proxy server for authentication so that the request can be processed.
408 Request Timeout	The request timed out.
409 Conflict	The request could not be processed due to a conflict.
500 Internal Server Error	Failed to complete the request because of a service error.
501 Not Implemented	Failed to complete the request because the server does not support the requested function.
502 Bad Gateway	Failed to complete the request because the request is invalid.
503 Service Unavailable	Failed to complete the request. The service is unavailable.
504 Gateway Timeout	A gateway timeout error occurred.

A.2 Error Codes

Status Code	Error Codes	Error Message	Description	Solution
400	TMS.0002	Bad request.	Invalid request data.	Check request parameters.
400	TMS.0007	Limit is invalid.	Invalid limit.	Specify a valid limit.
400	TMS.0008	Marker is invalid.	Invalid marker.	Specify a valid marker.
400	TMS.0009	Key is invalid.	Invalid key.	Specify a valid key.
400	TMS.0010	Value is invalid.	Invalid value.	Specify a valid value.
400	TMS.0011	Action is invalid.	Invalid action.	Specify a valid action.
400	TMS.0012	Tags is empty.	tags is left blank.	Specify a valid tags.

Status Code	Error Codes	Error Message	Description	Solution
400	TMS.0013	Empty element in tags.	Elements in tags are empty.	Specify a valid tags.
400	TMS.0016	Values is too much.	The maximum number of characters for value has been reached.	Specify a valid value.
400	TMS.0017	Offset is invalid.	Invalid offset.	Specify a valid tags.
400	TMS.1001	The number of predefine tag exceeds the upper limit.	The maximum number of predefined tags has been reached.	Delete unnecessary predefined tags.
400	TMS.1002	Old_tag cannot be found.	The old tag does not exist.	Check the old tag.
400	TMS.1003	New_tag already exists.	The new tag already exists.	Check the new tag.
400	TMS.1004	Old_tag is empty.	The old tag is left blank.	Check the old tag.
400	TMS.1005	Invalid key in old_tag.	Invalid key in the old tag.	Specify a valid key for the old tag.
400	TMS.1006	Invalid value in old_tag.	Invalid value in the old tag.	Specify a valid value for the old tag.
400	TMS.1007	New_tag is empty.	The new tag is left blank.	Check the new tag.
400	TMS.1008	Invalid key in new_tag.	Invalid key in the new tag.	Specify a valid key for the new tag.
400	TMS.1009	Invalid value in new_tag.	Invalid value in the new tag.	Specify a valid value for the new tag.
400	TMS.1010	Order_field is invalid.	Invalid sortField.	Specify a valid sortField.
400	TMS.1011	Order_method is invalid.	Invalid orderMethod.	Specify a valid orderMethod.

Status Code	Error Codes	Error Message	Description	Solution
401	TMS.0003	Unauthorized user.	Unauthorized request.	Check the authentication token.
403	TMS.0004	Permission error.	Insufficient permissions.	Check your permissions.
403	TMS.0006	The request is too much, try again later.	Too many requests.	Try again later.
404	TMS.0005	Requested resources not found.	Failed to find the resource.	Contact the service support personnel to check whether the API has been registered.
409	TMS.0014	Conflict	Internal conflict.	Contact service support personnel.
500	TMS.0001	System error.	System error.	Contact service support personnel.
504	TMS.0018	Query Time Out.	Query timed out.	Try again later.

A.3 Obtaining a Project ID

Scenarios

A project ID is required for some URLs when an API is called. Therefore, you need to obtain a project ID in advance. Two methods are available:

- **Obtain the Project ID by Calling an API**
- **Obtain the Project ID from the Console**

Obtain the 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 **Regions and Endpoints**. For details about API authentication, see **Authentication**.

The following is an example response. The value of **id** is the project ID.

```
{
    "projects": [
        {
            "domain_id": "65ewtrgaggshhk1223245sghjlse684b",
            "is_domain": false,
            "parent_id": "65ewtrgaggshhk1223245sghjlse684b",
            "name": "project_name",
            "description": "",
            "links": {
                "next": null,
                "previous": null,
                "self": "https://www.example.com/v3/projects/a4adasfjljaaaakla12334jklga9sasfg"
            },
            "id": "a4adasfjljaaaakla12334jklga9sasfg",
            "enabled": true
        }
    ],
    "links": {
        "next": null,
        "previous": null,
        "self": "https://www.example.com/v3/projects"
    }
}
```

Obtain a Project ID from the Console

To obtain a project ID from the console, perform the following operations:

1. Log in to the management console.
2. Click the username and select **My Credentials** from the drop-down list.

   On the **My Credentials** page, view the project ID (value in the **Project ID** column).

A.4 Obtaining the Domain-Level Token

The following example shows how to obtain the domain-level token.

```
POST https://{{endpoint}}/v3/auth/tokens
Content-Type: application/json

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

NOTE

- The validity period of a token is 24 hours. Cache the token to prevent frequent API calling.
- To avoid token expiration during an API call, ensure that the time taken to complete a call is shorter than the time left before a token expires.