Data Lake Visualization

API Reference

Issue 01

Date 2022-11-14

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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

Notice

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

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

Contents

1 Before You Start

1.1 Overview

Welcome to *Data Lake Visualization API Reference*. Data Lake Visualization (DLV) is a one-stop data visualization platform that adapts to various data sources in the cloud or on-premises. By dragging and dropping 2D and 3D visual components on DLV, you can quickly customize and use a data screen of your own.

This document describes how to use application programming interfaces (APIs) to perform operations on DLV, such as creating, deleting, or copying screens. For details about all supported operations, see **API Overview**.

If you plan to access DLV through an API, ensure that you are familiar with DLV concepts. For details, see **Service Overview**.

1.2 API Calling

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

1.3 Endpoints

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

1.4 Constraints

- The number of screens you can create is determined by your quota. For details, see **Service Quota**.
- For more constraints, see API description.

1.5 Concepts

- Account

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

- IAM User

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

  The account name, username, and password will be required for API authentication.

- Region

  Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections. For low network latency and quick resource access, select the nearest region.

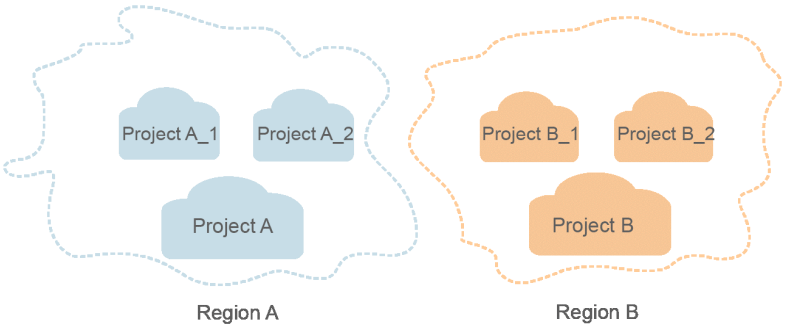
- AZ

  AZs are physically isolated locations in a region, but are interconnected through an internal network for enhanced application availability.

- Project

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

  Figure 1-1 Project isolating model

- Enterprise project

  Enterprise projects group and manage resources across regions. Resources in enterprise projects are logically isolated from each other. An enterprise project

can contain resources of multiple regions, and resources can be added to or removed from enterprise projects.

For more information about enterprise projects and how to obtain enterprise project IDs, see the **Enterprise Management User Guide**.

2 API Overview

The self-developed DLV APIs comply with REST API design specifications and enable you to use the functions listed in **Table 2-1**.

Table 2-1 API functions

Type	Function	URI
Screen management API	**Creating a Screen**	POST /v1/{project_id}/screens
	Deleting a Screen	DELETE /v1/{project_id}/screens/{screen_id}
	Changing a Screen Name	PUT /v1/{project_id}/screens/{screen_id}
	Obtaining a Screen List	GET /v1/{project_id}/screens
	Obtaining Screen Details	GET /v1/{project_id}/screens/{screen_id}
	Copying a Screen	POST /v1/{project_id}/screens/{screen_id}/copy
	Exporting a Screen	GET /v1/{project_id}/screen/{screen_id}/export
	Importing a Screen	POST /v1/{project_id}/screen/import?workspaceId={workspaceId}
Template management API	**Obtaining a Template List**	GET /v1/{project_id}/screens/templates

3 Calling APIs

3.1 Making an API Request

This section describes the structure of a REST API, and uses the IAM API for **obtaining a user token** as an example to demonstrate how to call an API. The obtained token is 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

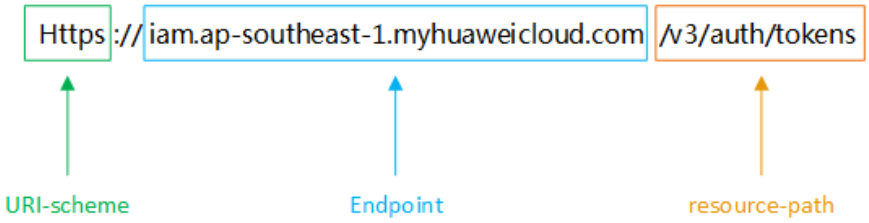
Parameter	Description
URI-scheme	Protocol used to transmit requests. All APIs use HTTPS.
Endpoint	Domain name or IP address of the server bearing the REST service. The endpoint varies between services in different regions. It can be obtained from **Regions and Endpoints**. For example, the endpoint of IAM in region **CN-Hong Kong** is **iam.ap-southeast-1.myhuaweicloud.com**.
resource-path	Access path of an API for performing a specified operation. Obtain the path from the URI of an API. For example, the **resource-path** of the API used to obtain a user token is **/v3/auth/tokens**.

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

For example, to obtain an IAM token in the **CN-Hong Kong** region, obtain the endpoint of IAM (**iam.ap-southeast-1.myhuaweicloud.com**) for this region and the **resource-path** (**/v3/auth/tokens**) in the URI of the API used to **obtain a user token**. Then, construct the URI as follows:

```
https://iam.ap-southeast-1.myhuaweicloud.com/v3/auth/tokens
```

Figure 3-1 Example URI



NOTE

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

Request Methods

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

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.

Method	Description
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://iam.ap-southeast-1.myhuaweicloud.com/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.

Table 3-3 lists common request header fields.

Table 3-3 Common request headers

Field	Description	Mandatory	Example
Host	Server information of the resource being requested. The value can be obtained from the URL of a service API. The value is **hostname[:port]**. Default port used for **https** requests is port **443**.	No This field is mandatory for AK/SK authentication.	code.test.com or code.test.com:443
Content-Type	Request body MIME type. This field is mandatory and its default value is **application/json**. Other values of this field will be provided for specific APIs if any.	Yes	application/json

Field	Description	Mandatory	Example
Content-Length	Length of the request body. The unit is byte.	No This field is mandatory for POST and PUT requests, but must be left blank for GET requests.	3495
X-Project-Id	Project ID. Obtain the project ID by following the instructions in **Obtaining a Project ID**.	No	e9993fc787d94b6c886cbaa340f9c0f4
X-Auth-Token	User token. It is a response to the API used to **obtain a user token**. This API is the only one that does not require authentication. After the request is processed, the value of **X-Subject-Token** in the header is the token value.	No This field is mandatory for token authentication.	The following is part of an example token: MIIPAgYJKoZIhvcNAQc-Co...ggg1BBIINPXsidG9rZ

NOTE

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

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

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

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json
```

(Optional) Request Body

The body of a request is often sent in a structured format as specified in the **Content-Type** header field, such as JSON or XML. 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*, ********* (login password), and *xxxxxxxxxxxxxxxxx* (project ID) with the actual values. To learn how to obtain a project ID, see **Obtaining a Project ID**.

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://iam.ap-southeast-1.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json

{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "username",
                    "password": "********",
                    "domain": {
                        "name": "domainname"
                    }
                }
            }
        },
        "scope": {
            "project": {
                "id": "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-based authentication: Requests are authenticated using a token.
- AK/SK-based authentication: Requests are authenticated by encrypting the request body using an AK/SK pair. AK/SK authentication is recommended because it is more secure than token authentication.

Token-based Authentication

NOTE

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

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.

When calling an API to **Obtain a User Token**, you must set **auth.scope** in the request body to **project**.

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

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

```
Content-Type: application/json
X-Auth-Token: ABCDEFJ....
```

AK/SK-based Authentication

NOTE

AK/SK-based authentication supports API requests with a body not larger than 12 MB. For API requests with a larger body, token-based authentication is recommended.

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

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

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

NOTICE

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

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 3-2 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.

Figure 3-2 Header fields of the response to the request for obtaining a user token

(Optional) Response Body

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

The following is 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": "ap-southeast-1",
...
```

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

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

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

4 Getting Started

This section describes the procedure for creating a screen by calling APIs:

1. If the token authentication is used, obtain the user token. The token will be put into the request header for authentication in a subsequent request.

2. Call an API to create a screen.

Prerequisite

- You have obtained the **endpoints of IAM** and **endpoints of DLV**.
- You have obtained project IDs. For details about project IDs, see **Obtaining a Project ID**.

Procedure

The following values are examples (replace them with actual values):

- IAM endpoint: **iam_endpoint**
- DLV endpoint: **dlv_endpoint**
- Project ID: **project_id**

Perform the following operations to create a screen:

Step 1 When using token authentication, obtain the user token and set it as an environment variable.

```
curl -H "Content-Type:application/json" https://iam_endpoint/v3/auth/tokens -X POST -d '{
   "auth": {
      "identity": {
         "methods": [
            "password"
         ],
         "password": {
            "user": {
               "name": "testname",
               "domain": {
                  "name": "testname"
               },
               "password": "Passw0rd"
            }
         }
      },
      "scope": {
         "project": {
```



```
            "name": "ap-southeast-1"
        }
    }
  }
}' -v -k
```

The value of **X-Subject-Token** in the response header is the token.

```
X-Subject-Token:MIIDkgYJKoZIhvcNAQcCoIIDgzCCA38CAQExDTALBglghkgBZQMEAgEwgXXXXX...
```

Run the following command to set the token as an environment variable for future use: The value of **X-Auth-Token** is the token value obtained in the previous step.

```
export Token=MIIDkgYJKoZIhvcNAQcCoIIDgzCCA38CAQExDTALBglghkgBZQMEAgEwgXXXXX...
```

Step 2 Call the API for creating a screen.

```
curl -X POST -H 'Content-Type:application/json;charset=utf-8' -H "X-Auth-Token:$Token" -d '{
   "name": "my_screen",
   "alias": "first_screen",
   "description": "my first screen",
   "template_id": "453113551",
   "workspaceId":"86ce107974ce4f93b618acb232863027"
   }
}' https://dlv_endpoint/v1/project_id/screens -v -k
```

Screen parameters are described as follows:

- **name**: name of the screen.
- **alias**: alias of the screen.
- **description**: description of the screen.
- **template_id**: ID of the screen template.
- **workspaceId**: workspace ID.

If status code **200** is returned, the screen is successfully created. If a status code listed **Status Codes** is returned, the screen fails to be created.

----End

5 DLV APIs

5.1 Screen Management

5.1.1 Creating a Screen

Function

Create a screen. The screen content comes from a screen template.

URI

- URI format
  POST /v1/{project_id}/screens
- Parameter description

Table 5-1 URI parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request

Table 5-2 describes the request parameters.

Table 5-2 Request parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Name of the screen. The value contains 1 to 32 characters and consists of digits and letters.

Parameter	Mandatory	Type	Description
alias	No	String	Alias of the screen. The value contains 1 to 255 characters. The default value is empty.
description	No	String	Description of the screen. The value is 0 to 512 characters. The default value is empty.
templateId	Yes	String	Template ID of the screen. For details about how to obtain the template ID, see **Obtaining a Template List**.
workspaceId	Yes	String	Workspace ID, which consists of 32 characters. For details about how to obtain the workspace ID, see **Obtaining a Workspace ID**.

Response

Table 5-3 describes the response parameters.

Table 5-3 Response parameters

Parameter	Type	Description
id	String	ID of the screen.
name	String	Name of the screen.
alias	String	Alias of the screen.
description	String	Description of the screen.
status	Integer	Status of the screen: 0-created, 1-deleted, 2-shared
projectId	String	ID of the project.
createUser	String	Creator of the screen.
createDate	Integer	Timestamp for creating the screen.
updateUser	String	User who updates the screen.
updateDate	Integer	Timestamp for updating the screen.
templateId	String	Template ID used when creating the screen.
snapshotId	String	ID of the screen snapshot.
share	String	UUID for sharing the screen.
thumbnail	String	URL for accessing the screen thumbnail.
config	**config** object	Configuration content of the screen.

Table 5-4 config data structure

Parameter	Type	Description
scenes	Array of **scene** objects	Scene information.
config	Array of **config** objects	Configuration details.

Table 5-5 scene

Parameter	Type	Description
name	String	Name of the scene.
layers	Array of **layer** objects	Layers of the scene.

Table 5-6 layer

Parameter	Type	Description
id	String	ID of the layer.
name	String	Name of the layer.
coms	Array	Component list.

Table 5-7 config

Parameter	Type	Description
id	String	Unique ID of the screen.
screenId	String	ID of the screen.
grid	Integer	Grid spacing in units of pixels.
backgroundImage	String	OBS link of the background image.
backgroundColor	String	Color of the background.
width	Integer	Screen width in units of pixels.
height	Integer	Screen height in units of pixels.
whLinkage	Integer	Association of width and height. 0: Disable; 1: Enable

Parameter	Type	Description
comList	String	Component list.
screenshot	String	Cover image.
display	Integer	Zoom modes: • 1. Uniform scaling with width • 2. Uniform scaling with height • 3. Uniform adaptive scaling • 4: Full screen stretch
watermarkFlag	Integer	Specifies whether a watermark is available.
rulerLines	**rulerLines** object	Coordinate.
variables	**variables** object	Interaction information.
workspaceId	String	Workspace ID.

Table 5-8 rulerLines

Parameter	Type	Description
h	String	Horizontal ruler line coordinate in units of pixels.
v	String	Vertical ruler line coordinate in units of pixels.
show	Boolean	Specifies whether to display ruler lines. The options are **true** (yes) and **false** (no).

Table 5-9 variables

Parameter	Type	Description
publishersView	String	Interaction parameters of the sending component.
subscribersView	String	Interaction parameters of the receiving component.

Examples

Create **screen_01**. The template ID is 32546212564.

- Request example

```
POST https://{dlv_endpoint}/v1/{project_id}/screens
Request header

{
   "name":"screen_01",
   "alias":"test_screen_01",
   "description":"for test",
   "templateId":"32546212564",
   "workspaceId":"86ce107974ce4f93b618acb232863027"
}
```

- Example of a successful response

```
{
   "id": "ff80808167813a360167819d5b640045",
   "name": "screen_01",
   "alias": "test_screen_01",
   "description": "for test",
   "status": 0,
   "projectId": "abcdefghijk0123456",
   "createUser": "a1b2c2d3e4f5g6h5j6k6",
   "createDate": 1544067832676,
   "updateUser": null,
   "updateDate": null,
   "templateId": "2a8281f9678bb29f01678bb9b2d60001",
   "snapshotId": null,
   "share": null,
   "thumbnail": null,
   "config": {
      "scenes": [
         {
"name": "scenario 1",
            "layers": [
               {
                  "id": null,
"name": "layer 1",
                  "coms": []
               }
            ]
         }
      ],
      "config": {
         "id": null,
         "screenId": "ff80808167813a360167819d5b640045",
         "grid": null,
         "backgroundImage": "",
         "backgroundColor": null,
         "width": 1920,
         "height": 1080,
         "whLinkage": null,
         "comList": "",
         "screenshot": null,
         "display": 1,
         "watermarkFlag": null,
         "rulerLines": null,
         "variables": null
      }
   }
}
```

- Example of a failed response

```
{
   "errors": [
      {
         "error_code": "DLV.1011",
         "error_msg": "Invalid template id."
      }
   ]
}
```


Status Codes

For details about status codes, see **Status Codes**.

5.1.2 Deleting a Screen

Function

Delete a screen.

URI

- URI format
  DELETE /v1/{project_id}/screens/{screen_id}
- Parameter description

Table 5-10 URI parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.
screen_id	Yes	String	Screen ID. For details about how to obtain the screen ID, see **Obtaining a Screen List**.

Request

Table 5-11 describes the request parameters.

Table 5-11 Request parameters

Parameter	Mandatory	Type	Description
workspaceId	Yes	String	Workspace ID, which consists of 32 characters. For details about how to obtain the workspace ID, see **Obtaining a Workspace ID**.

Response

Table 5-12 describes the response parameters.

Table 5-12 Response parameters

Parameter	Type	Description
is_success	Boolean	Specifies whether the operation is successful. **true**: The operation is successful. **false**: The operation failed.
statusCode	Integer	Status code.
message	String	Message returned.

Example

Delete the screen whose ID is ff80808167813a360167819d5b640045.

- Request example
```
DELETE https://{dlv_endpoint}/v1/{project_id}/screens/ff80808167813a360167819d5b640045
Request header

{
    "workspaceId": "86ce107974ce4f93b618acb232863027"
}
```
- Example of a successful response
```
{
    "is_success": true,
    "statusCode": 200,
    "message": null
}
```
- Example of a failed response
```
{
    "errors": [
        {
            "error_code": "1004",
            "error_msg": "The operation with the resource entity occur some error."
        }
    ]
}
```

Status Codes

For details about status codes, see **Status Codes**.

5.1.3 Changing a Screen Name

Function

Change a screen name.

URI

- URI format
```
PUT /v1/{project_id}/screens/{screen_id}
```
- Parameter description

Table 5-13 URI parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.
screen_id	Yes	String	Screen ID. For details about how to obtain the screen ID, see **Obtaining a Screen List**.

Request

Table 5-14 describes the request parameters.

Table 5-14 Request parameters

Parameter	Mandatory	Type	Description
name	Yes	String	Name of the screen. A screen name contains 1 to 32 characters and consists of digits and letters.
workspaceId	Yes	String	Workspace ID, which consists of 32 characters. For details about how to obtain the workspace ID, see **Obtaining a Workspace ID**.

Response

Table 5-15 describes the response parameters.

Table 5-15 Response parameters

Parameter	Type	Description
is_success	Boolean	Specifies whether the operation is successful. **true**: The operation is successful. **false**: The operation failed.
statusCode	Integer	Status codes.
message	String	Message returned.

Examples

Change the name of the screen whose ID is ff80808167813a360167819d5b640045 to **screen_01_change**.

- Request example

```
PUT https://{dlv_endpoint}/v1/{project_id}/screens/ff80808167813a360167819d5b640045
Request header

{
   "name":"screen_01_change",
   "workspaceId":"86ce107974ce4f93b618acb232863027"}
```

- Example of a successful response

```
{
   "is_success": true,
   "statusCode": 200,
   "message": null
}
```

- Example of a failed response

```
{
   "errors": [
      {
         "error_code": "1004",
         "error_msg": "The operation with the resource entity occur some error."
      }
   ]
}
```

Status Codes

For details about status codes, see **Status Codes**.

5.1.4 Obtaining a Screen List

Function

Obtain a screen list.

URI

- URI format

```
GET /v1/{project_id}/screens
```

- Parameter description

Table 5-16 URI parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request

Table 5-17 describes the request parameters.

Table 5-17 Request parameters

Parameter	Mandatory	Type	Description
workspaceId	Yes	String	Workspace ID, which consists of 32 characters. For details about how to obtain the workspace ID, see **Obtaining a Workspace ID**.

Response

Table 5-18 describes the response parameters.

Table 5-18 Response parameters

Parameter	Type	Description
screens	Array of **screen** objects	Screen list.
maxScreenNuM	Integer	The maximum number of the screens that can be created.
currentNum	Integer	Number of screens created in the current project.
domainScreenNum	Integer	Number of screens created for all projects in the current region.

Table 5-19 screen

Parameter	Type	Description
id	String	ID of the screen.
name	String	Name of the screen.
alias	String	Alias of the screen.
description	String	Description of the screen.
status	Integer	Status of the screen: 0-created, 1-deleted, 2-shared
projectId	String	ID of the project.
createUser	String	Creator of the screen.
createDate	Integer	Timestamp for creating the screen.
updateUser	String	User who updates the screen.

Parameter	Type	Description
updateDate	Integer	Timestamp for updating the screen.
templateId	String	ID of the template used for creating the screen.
snapshotId	String	ID of the screen snapshot.
share	String	UUID for sharing the screen.
thumbnail	String	URL for accessing the screen thumbnail.
width	Integer	Screen width in units of pixels.
height	Integer	Screen height in units of pixels.
password	String	Password for accessing the screen
token	String	Token for accessing the screen.

Examples

- Request example

  GET https://*{dlv_endpoint}*/v1/*{project_id}*/screens
  Request header

  ```
  {
     "workspaceId": "86ce107974ce4f93b618acb232863027"
  }
  ```

- Example of a successful response

  ```
  {
     "screens": [
        {
           "id": "ff80808167813a360167819d5b640045",
           "name": "screen_01",
           "alias": "test_screen_01",
           "description": null,
           "status": 2,
           "projectId": "abcdefghijk0123456",
           "createUser": "a1b2c2d3e4f5g6h5j6k6",
           "createDate": 1544067832676,
           "updateUser": null,
           "updateDate": 0,
           "templateId": "32546212564",
           "snapshotId": null,
           "share": "f7c9a336e6c74ca5883a60af882d92de",
           "thumbnail": null,
           "width": 1920,
           "height": 1080,
           "password": null,
           "token": null
        },
        {
           "id": "ff808081674bb2e401675002b5950120",
           "name": "Test_done",
           "alias": "",
           "description": null,
           "status": 2,
           "projectId": "abcdefghijk0123456",
           "createUser": "a1b2c2d3e4f5g6h5j6k6",
           "createDate": 1543235614101,
           "updateUser": "a1b2c2d3e4f5g6h5j6k6",
           "updateDate": 1543238083522,
  ```



```
            "templateId": "32546212564",
            "snapshotId": null,
            "share": "e3f93ab1474f43ff8e45ddaa90157d30",
            "thumbnail": null,
            "width": 1920,
            "height": 1080,
            "password": null,
            "token": null
        }

    ],
    "maxScreenNuM": 20,
    "currentNum": 2,
    "domainScreenNum": 2
}
```


- Example of a failed response


```
{
    "errors": [
        {
            "error_code": "1004",
            "error_msg": "The operation with the resource entity occur some error."
        }
    ]
}
```


Status Codes

For details about status codes, see **Status Codes**.

5.1.5 Obtaining Screen Details

Function

Obtain screen details.

URI

- URI format

```
GET /v1/{project_id}/screens/{screen_id}
```

- Parameter description

Table 5-20 URI parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.
screen_id	Yes	String	Screen ID. For details about how to obtain the screen ID, see **Obtaining a Screen List**.

Request

Table 5-21 describes the request parameters.

Table 5-21 Request parameters

Parameter	Mandatory	Type	Description
workspaceId	Yes	String	Workspace ID, which consists of 32 characters. For details about how to obtain the workspace ID, see **Obtaining a Workspace ID**.

Response

Table 5-22 describes the response parameters.

Table 5-22 Response parameters

Parameter	Type	Description
id	String	ID of the screen.
name	String	Name of the screen.
alias	String	Alias of the screen.
description	String	Description of the screen.
status	Integer	Status of the screen: 0-created, 1-deleted, 2-shared
projectId	String	ID of the project.
createUser	String	Creator of the screen.
createDate	Integer	Timestamp for creating the screen.
updateUser	String	User who updates the screen.
updateDate	Integer	Timestamp for updating the screen.
templateId	String	ID of the template used for creating the screen.
snapshotId	String	ID of the screen snapshot.
share	String	UUID for sharing the screen.
thumbnail	String	URL for accessing the screen thumbnail.
config	**config** object	Configuration content of the screen.

Table 5-23 config data structure

Parameter	Type	Description
scenes	Array of **scene** objects	Scene information.

Parameter	Type	Description
config	Array of **config** objects	Configuration details.

Table 5-24 scene

Parameter	Type	Description
name	String	Name of the scene.
layers	Array of **layer** objects	Layers of the scene.

Table 5-25 layer

Parameter	Type	Description
id	String	ID of the layer.
name	String	Name of the layer.
coms	Array	Component list.

Table 5-26 config

Parameter	Type	Description
id	String	Unique ID of the screen.
screenId	String	ID of the screen.
grid	Integer	Grid spacing in units of pixels.
backgroundImage	String	OBS link of the background image.
backgroundColor	String	Color of the background.
width	Integer	Screen width in units of pixels.
height	Integer	Screen height in units of pixels.
whLinkage	Integer	Association of width and height. 0: Disable; 1: Enable
comList	String	Component list.
screenshot	String	Cover image.

Parameter	Type	Description
display	Integer	Zoom modes: • 1. Uniform scaling with width • 2. Uniform scaling with height • 3. Uniform adaptive scaling • 4: Full screen stretch
watermarkFlag	Integer	Specifies whether a watermark is available.
rulerLines	**rulerLines** object	Coordinate.
variables	**variables** object	Interaction information.
workspaceId	String	Workspace ID.

Table 5-27 rulerLines

Parameter	Type	Description
h	String	Horizontal ruler line coordinate in units of pixels.
v	String	Vertical ruler line coordinate in units of pixels.
show	Boolean	Specifies whether to display ruler lines. The options are **true** (yes) and **false** (no).

Table 5-28 variables

Parameter	Type	Description
publishersView	String	Interaction parameters of the sending component.
subscribersView	String	Interaction parameters of the receiving component.

Examples

Obtain the details about the screen whose ID is ff80808167813a360167819d5b640045.

- Request example
  GET https://*{dlv_endpoint}*/v1/*{project_id}*/screens/ff80808167813a360167819d5b640045
  Request header


```
{
    "workspaceId": "86ce107974ce4f93b618acb232863027"
}
```

- Example of a successful response

```
{
    "id": "ff80808167813a360167819d5b640045",
    "name": "screen_01",
    "alias": "test_screen_01",
    "description": "for test",
    "status": 2,
    "projectId": "abcdefghijk0123456",
    "createUser": "a1b2c2d3e4f5g6h5j6k6",
    "createDate": 1544067832676,
    "updateUser": null,
    "updateDate": null,
    "templateId": "32546212564",
    "snapshotId": null,
    "share": "f7c9a336e6c74ca5883a60af882d92de",
    "thumbnail": null,
    "config": {
        "scenes": [
            {
"name": "scenario 1",
                "layers": [
                    {
                        "id": "ff80808167813a360167819d5b640047",
"name": "layer 1",
                        "coms": []
                    }
                ]
            }
        ],
        "config": {
            "id": "ff80808167813a360167819d5b650048",
            "screenId": "ff80808167813a360167819d5b640045",
            "grid": null,
            "backgroundImage": "",
            "backgroundColor": null,
            "width": 1920,
            "height": 1080,
            "whLinkage": null,
            "comList": "",
            "screenshot": null,
            "display": 1,
            "watermarkFlag": null,
            "rulerLines": null,
            "variables": null
        }
    }
}
```

- Example of a failed response

```
{
    "errors": [
        {
            "error_code": "1004",
            "error_msg": "The operation with the resource entity occur some error."
        }
    ]
}
```

Status Codes

For details about status codes, see **Status Codes**.

5.1.6 Copying a Screen

Function

Copy a screen.

URI

- URI format
  POST /v1/{project_id}/screens/{screen_id}/copy
- Parameter description

Table 5-29 URI parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.
screen_id	Yes	String	ID of the screen to be copied. For details about how to obtain the ID, see **Obtaining a Screen List**.

Request

Table 5-30 describes the request parameters.

Table 5-30 Request parameters

Parameter	Mandatory	Type	Description
workspaceId	Yes	String	Workspace ID, which consists of 32 characters. For details about how to obtain the workspace ID, see **Obtaining a Workspace ID**.

Response

Table 5-31 describes the response parameters.

Table 5-31 Response parameter description

Parameter	Type	Description
id	String	ID of the screen.
name	String	Name of the screen.

Parameter	Type	Description
alias	String	Alias of the screen.
description	String	Description of the screen.
status	Integer	Status of the screen: 0-created, 1-deleted, 2-shared
projectId	String	ID of the project.
createUser	String	Creator of the screen.
createDate	Integer	Timestamp for creating the screen.
updateUser	String	User who updates the screen.
updateDate	Integer	Timestamp for updating the screen.
templateId	String	ID of the template used for creating the screen.
snapshotId	String	ID of the screen snapshot.
share	String	UUID for sharing the screen.
thumbnail	String	URL for accessing the screen thumbnail.
config	JSON	Configuration content of the screen.

Examples

- Request example

  POST https://*{dlv_endpoint}*/v1/*{project_id}*/screens/ff80808167813a360167819d5b640045/copy
  Request header

  ```
  {
     "workspaceId": "86ce107974ce4f93b618acb232863027"
  }
  ```

- Example of a successful response

  ```
  {
     "id": "ff80808167813a360167825a27b00073",
     "name": "screen_01_copy",
     "alias": "test_screen_01",
     "description": "for test",
     "status": 0,
     "projectId": "abcdefghijk0123456",
     "createUser": "a1b2c2d3e4f5g6h5j6k6",
     "createDate": 1544080205744,
     "updateUser": null,
     "updateDate": null,
     "templateId": "32546212564",
     "snapshotId": null,
     "share": null,
     "thumbnail": null,
     "config": null
  }
  ```

- Example of a failed response

  ```
  {
     "errors": [
        {
           "error_code": "1004",
           "error_msg": "The operation with the resource entity occur some error."
  ```



```
    }
  ]
}
```

Status Codes

For details about status codes, see **Status Codes**.

5.1.7 Exporting a Screen

Function

Export a specified screen.

URI

- URI format
  GET /v1/{project_id}/screen/{screen_id}/export
- Parameter description

Table 5-32 URI parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.
screen_id	Yes	String	ID of the screen to be exported. For details about how to obtain the ID, see **Obtaining a Screen List**.

Request

Table 5-33 describes the request parameters.

Table 5-33 Request parameters

Parameter	Mandatory	Type	Description
workspaceId	Yes	String	Workspace ID, which consists of 32 characters. For details about how to obtain the workspace ID, see **Obtaining a Workspace ID**.

Response

Table 5-34 describes the response parameters.

Table 5-34 Parameter description

Parameter	Type	Description
id	String	ID of the screen.
name	String	Name of the screen.
alias	String	Alias of the screen.
description	String	Description of the screen.
status	Integer	Status of the screen: 0-created, 1-deleted, 2-shared
projectId	String	ID of the project.
createUser	String	Creator of the screen.
createDate	Integer	Timestamp for creating the screen.
updateUser	String	User who updates the screen.
updateDate	Integer	Timestamp for updating the screen.
templateId	String	ID of the template used for creating the screen.
snapshotId	String	ID of the screen snapshot.
share	String	UUID for sharing the screen.
thumbnail	String	URL for accessing the screen thumbnail.
config	JSON	Configuration content of the screen.

Example

- Request example
  ```
  GET https://{dlv_endpoint}/v1/{project_id}/screens/{screen_id}/export?
  workspaceId=86ce107974ce4f93b618acb232863027{Request header}
  ```
- Example of a successful response
  ```
  {
     Binary stream
  }
  ```
- Example of a failed response
  ```
  {
     "errors": [
        {
           "error_code": "1004",
           "error_msg": "The operation with the resource entity occur some error."
        }
     ]
  }
  ```

Status Code

For details about status codes, see **Status Codes**.

5.1.8 Importing a Screen

Function

Import the exported screen to DLV. Before importing a screen, you can call the export API to export the required screen. For details about how to export a screen, see **Exporting a Screen**. The exported screen file can be imported when required.

URI

- URI format
  POST /v1/{project_id}/screen/import?workspaceId={workspaceId}
- Parameter description

  Table 5-35 URI parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.
workspaceId	Yes	String	Workspace ID, which consists of 32 characters. For details about how to obtain the workspace ID, see **Obtaining a Workspace ID**.

Request

Table 5-36 describes the request parameters.

Table 5-36 Request parameters

Parameter	Mandatory	Type	Description
file	Yes	String	Screen file to be imported. The file is in ***.zip** format. You can call the export API to export a screen. The exported **.zip** file can be imported to DLV.

Response

Table 5-37 describes the request parameters.

Table 5-37 Parameter description

Parameter	Type	Description
is_success	boolean	Specifies whether the import succeeds or not.
statusCode	int	Status code.
message	string	Response message.

Example

- Request example
```
POST https://{dlv_endpoint}/v1/{project_id}/screen/import?
workspaceId=86ce107974ce4f93b618acb232863027
Request header

{
    file:xxx.zip
}
```
- Example of a successful response
```
{
    "is_success": true,
    "statusCode": 200,
    "message": null
}
```

Status Code

For details about status codes, see **Status Codes**.

5.2 Template Management

5.2.1 Obtaining a Template List

Function

Obtain a template list.

URI

- URI format
```
GET /v1/{project_id}/screens/templates
```
- Parameter description

Table 5-38 URI parameter description

Parameter	Mandatory	Type	Description
project_id	Yes	String	Project ID. For details about how to obtain the project ID, see **Obtaining a Project ID**.

Request

Table 5-39 describes the request parameters.

Table 5-39 Request parameters

Parameter	Mandatory	Type	Description
workspaceId	Yes	String	Workspace ID, which consists of 32 characters. For details about how to obtain the workspace ID, see **Obtaining a Workspace ID**.

Response

Table 5-40 describes the response parameters.

Table 5-40 Response parameters

Parameter	Type	Description
id	String	ID of the template.
name	String	Name of the template.
description	String	Description of the template.
thumbnail	String	Address for storing template thumbnails.
priority	Integer	Template priority, that is, the template position on the console. The first digit from left to right is 0.
size	String	Size of the template screen.
createDate	Long	Time when the template was created.
updateDate	Long	Time when the template was updated.
status	Integer	Status of the screen: 0-created, 1-deleted, 2-shared
locale	String	Language identifier.

Parameter	Type	Description
permission	Boolean	Indicates whether the user has the permission to access the template. **true**: The user has permission. **false**: The user has no permission.

Examples

- Request example

```
GET https://{dlv_endpoint}/v1/{project_id}/screens/templates
Request header

{
   "workspaceId": "86ce107974ce4f93b618acb232863027"
}
```

- Example of a successful response

```
[
   {
      "id": "32546212564",
      "name": "blank_template",
"description": "create a screen of your own on the blank canvas." ,
      "thumbnail": "",
      "priority": 0,
"Size": "user-defined",
      "createDate": 1533176653819,
      "updateDate": 1533176653819,
      "status": 0,
      "locale": en-us,
      "permission": true
   }
]
```

- Example of a failed response

```
{
   "errors": [
      {
         "error_code": "1004",
         "error_msg": "The operation with the resource entity occur some error."
      }
   ]
}
```

Status Codes

For details about status codes, see **Status Codes**.

6 Appendix

6.1 Status Codes

Table 6-1 lists all status codes.

Table 6-1 Status codes

Status Code	Reason Phrase	Description
200	OK	The request has been fulfilled.
201	Created	The request has been fulfilled and a new resource has been created.
202	Accepted	The request has been accepted, but the processing has not been completed.
204	NoContent	The request has been fulfilled, but the HTTP response does not contain a response body. The status code is returned in response to an HTTP OPTIONS request.
300	Multiple Choices	There are multiple options for the location of the requested resource. The response contains a list of resource characteristics and addresses from which the user or user agent (such as a browser) can choose the most appropriate one.
303	See Other	The response to the request can be found under a different URI, and should be retrieved using a GET or POST method.
304	Not Modified	The requested resource has not been modified. When the server returns this status code, it does not return any resources.

Status Code	Reason Phrase	Description
400	BadRequest	The request is invalid. The client should not repeat the request without modifications.
401	Unauthorized	The status code is returned after the client provides the authentication information, indicating that the authentication information is incorrect or invalid.
403	Forbidden	The server understood the request, but is refusing to fulfill it. The client should not repeat the request without modifications.
404	NotFound	The requested resource cannot be found. The client should not repeat the request without modifications.
406	Not Acceptable	The server cannot fulfill the request according to the content characteristics of the request.
409	Conflict	The request could not be processed due to a conflict with the current state of the resource. This status code indicates that the resource that the client is attempting to create already exists, or that the request has failed to be processed because of the update of the conflict request.
410	Gone	The requested resource is no longer available. The status code indicates that the requested resource has been deleted permanently.
412	Precondition Failed	The server did not meet one of the preconditions that the requester put on the request.
415	Unsupported Media Type	The server is unable to process the media format in the request.
500	InternalServerError	The server is able to receive the request but unable to understand it.
503	ServiceUnavailable	The requested service is invalid. The client should not repeat the request without modifications.

6.2 Error Codes

If API calling fails, no result data is returned. You can locate the error cause according to the error code of each API. When the API calling fails, HTTP status code 4*xx* or 5*xx* is returned. The returned message body contains the specific error code and error information. If you fail to locate the cause of an error, contact customer service and provide the error code so that we can help you solve the problem as soon as possible.

The following is an example of an error response:

```
{
   "errors": [
      {
         "error_code": "DLV.1000",
         "error_msg": "Screen number exceed quota."
      }
   ]
}
```

Table 6-2 Error codes

Status Code	Error Code	Error Message	Solution
400	DLV.1000	Insufficient screen quota.	Delete the screen and try again.
400	DLV.1001	The screen name exists. Use another name.	Check whether the name of the screen is the same as that of the existing screen.
400	DLV.1003	Processing failed.	Contact the service administrator.
400	DLV.1004	Errors occurred during your operation.	Contact the service administrator.
400	DLV.1005	Invalid screen name.	Correct the screen name and try again.
500	DLV.1006	Invalid screen configuration.	Correct the request parameter settings and try again.
400	DLV.1007	The screen does not exist.	Check the screen ID and ensure it is correct.
400	DLV.1008	The screen is already shared.	Check whether the screen has been published.
400	DLV.1009	The screen is not shared.	Publish the screen and try again.

Status Code	Error Code	Error Message	Solution
400	DLV.1010	The component configuration is null.	Correct the request parameter settings and try again.
400	DLV.1011	Invalid template ID.	Correct the template ID and try again.
400	DLV.1012	Invalid screen ID.	Correct the screen ID and try again.
400	DLV.1013	Invalid snapshot ID.	Correct the snapshot ID and try again.
400	DLV.1014	The data source name exists. Use another name.	Change the name and try again.
400	DLV.1015	The data source name does not exist.	Correct the parameter settings and try again.
400	DLV.1016	The parameter information is incomplete.	Correct the parameter settings and try again.
400	DLV.1017	The data source type is not supported.	Change the supported data source type and try again.
400	DLV.1018	The component does not exist.	Correct the parameter settings and try again.
400	DLV.1019	Incorrect subordinate relationship.	Correct the parameter settings and try again.
400	DLV.1020	The template ID cannot be empty.	Correct the parameter settings and try again.
400	DLV.1021	Insufficient snapshot quota.	Ensure that the number of snapshots does not exceed the threshold and try again.
500	DLV.1022	The comList of screenConfig mismatches the components in the component list.	Correct the parameter settings and try again.
400	DLV.1023	Failed to obtain the cluster.	Contact the service administrator.
400	DLV.1024	The image does not exist.	Correct the parameter settings and try again.
400	DLV.1025	Failed to create the data connection.	Contact the service administrator.

Status Code	Error Code	Error Message	Solution
400	DLV.1026	Invalid template name.	Correct the parameter settings and try again.
400	DLV.1027	The template is already published.	Do not publish the template again.
400	DLV.1028	The template is not published.	Publish the template and try again.
400	DLV.1029	The template does not exist.	Correct the parameter settings and try again.
400	DLV.1030	The template status is null.	Contact the service administrator.
400	DLV.1031	Failed to submit the job.	Contact the service administrator.
400	DLV.1032	Failed to obtain the job status.	Contact the service administrator.
400	DLV.1033	Failed to obtain the job result.	Contact the service administrator.
400	DLV.1034	Failed to connect to DLI.	Contact the service administrator.
400	DLV.1035	Failed to obtain the job result because the schema or row is empty.	Contact the service administrator.
400	DLV.1036	Failed to obtain the queue.	Contact the service administrator.
400	DLV.1037	Failed to obtain the database.	Contact the service administrator.
400	DLV.1038	Failed to upload the image.	Correct the parameter settings and try again.
400	DLV.1039	The file type is incorrect.	Correct the parameter settings and try again.
400	DLV.1040	The file size exceeds the upper limit.	Try a smaller file.
400	DLV.1041	Invalid data source name.	Correct the parameter settings and try again.
400	DLV.6311	Failed to obtain the MRS cluster information.	Contact the service administrator.

6.3 Obtaining a Project ID

Obtaining a Project ID by Calling an API

You can obtain the project ID by calling the 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 a response example. In the response, id indicates the project ID in the region corresponding to **name**.

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

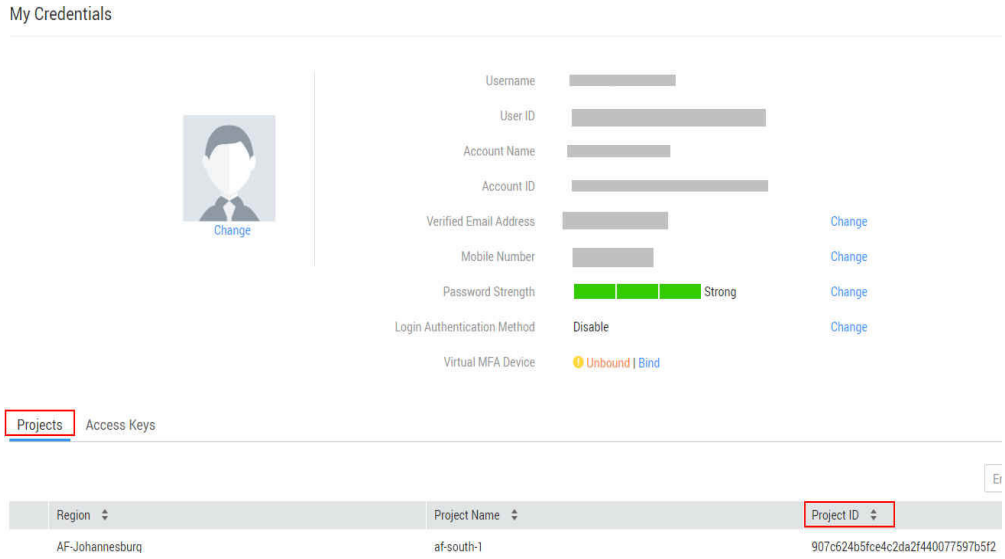
Obtaining a Project ID from the Management Console

When calling APIs, you need to specify the project ID in certain URLs. To do so, you need to obtain the project ID first. To obtain a project ID, perform the following operations:

1. Log in to the management console.
2. Hover the mouse over the username and select **Basic Information**.
3. On the displayed page, click **Manage** in **Security Credentials**.

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

Figure 6-1 Viewing project IDs

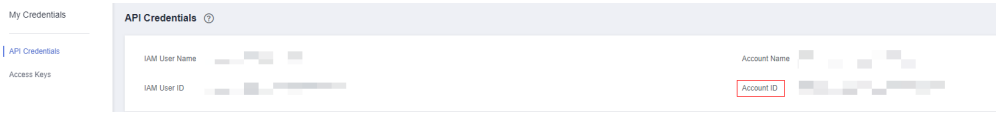


6.4 Obtaining an account ID

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

1. Log in to the management console after registration.
2. Click the username and select **My Credentials** from the drop-down list.
   On the **API Credentials** page, view **Account ID**.

Figure 6-2 Obtaining an account ID



6.5 Obtaining a Workspace ID

This section describes how to obtain a workspace ID.

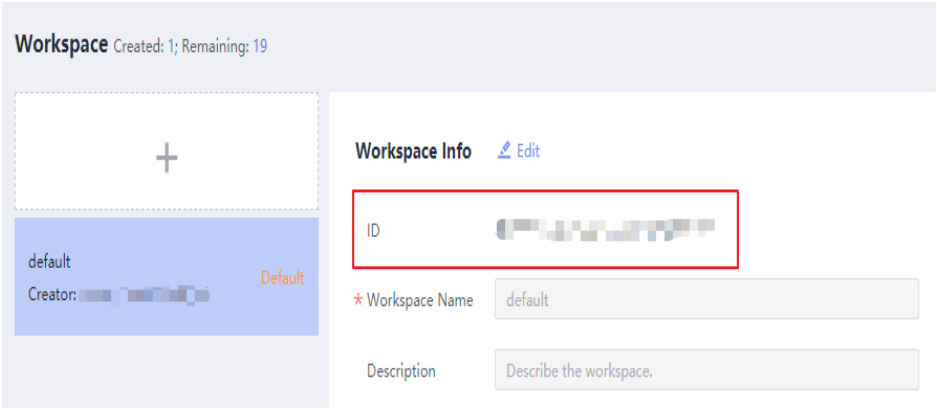
Obtaining a Workspace ID

Step 1 Log in to the DLV console.

Step 2 In the navigation tree on the left, choose **Control Center > Workspace**. The **Workspace** page is displayed.

Step 3 In the workspace list on the left, click the desired workspace and copy the value of **ID**.

Figure 6-3 Selecting a workspace



----End

A Change History

Release Date	Description
2020-08-14	This issue is the first official release.

