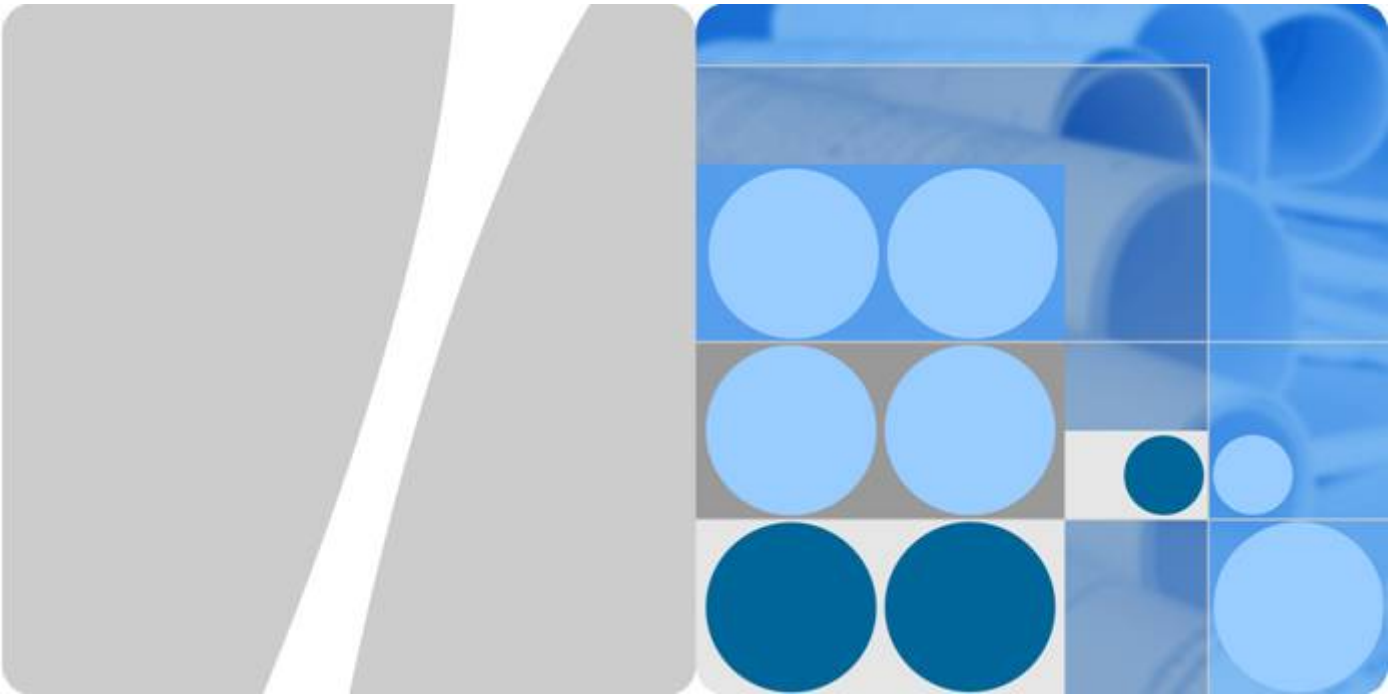

Data Ingestion Service

API Reference

Date **2018-10-25**

Contents

1 Before You Start

About This Document

This document provides a description of the Data Ingestion Service (DIS) API, parameter descriptions, and examples of syntax and usage. **Table 1-1** can help you find the information you are looking for.

To learn the terms used in this document, such as APP and checkpoint, see **Basic Terms**.

Table 1-1 Document overview

If You Want to Know	Go To
API components and lists	**API Overview**
Preparations before using the API	**Authenticating API Requests**
Components, calling methods, and examples of the Representational State Transfer (REST) body	● **REST API Overview** ● **API Calling Process**
API description	**Creating a DIS Stream~~Tag Management APIs**
Common parameters	● **Common Request Message Headers** ● **Common Response Message Headers** ● **Error Codes** ● **Status Codes**

2 API Overview

The DIS API is a self-developed API that complies with REST API design specifications. DIS provides the functions listed in **Table 2-1** through the DIS API.

Table 2-1 API functions

API	Function	API URI
Stream management	**Creating a DIS Stream**	POST /v2/{project_id}/streams
	Deleting a DIS Stream	POST /v2/{project_id}/streams
	Listing DIS Streams	GET /v2/{project_id}/streams
	Viewing Details of a DIS Stream	GET /v2/{project_id}/streams/{stream_name}
Data management	**Putting Data into a DIS Stream**	POST /v2/{project_id}/records
	Pulling Data from a DIS Stream	GET /v2/{project_id}/records{?partition-cursor}
	Obtaining a Cursor	GET /v2/{project_id}/cursors{?stream-name,partition-id,cursor-type,starting-sequence-number}
Program management	**Creating a Consumer Application**	POST /v2/{project_id}/apps
	Deleting a Consumer Application	DELETE /v2/{project_id}/apps/{app_name}
Checkpoint management	**Adding a Checkpoint**	POST /v2/{project_id}/checkpoints
	Querying a Checkpoint	GET /v2/{project_id}/checkpoints{?stream_name,partition_id,app_name,checkpoint_type}

API	Function	API URI
Tag management	**Adding a Tag to a Specified Stream**	POST /v1.1/{project_id}/stream/{stream_id}/tags
	Deleting a Tag of a Specified Stream	DELETE /v1.1/{project_id}/stream/{stream_id}/tags/{key}
	Querying a Tag of a Specified Stream	GET /v1.1/{project_id}/stream/{stream_id}/tags
	Adding or Deleting Stream Tags in Batches	POST /v1.1/{project_id}/stream/{stream_id}/tags/action
	Querying All Tags	GET /v1.1/{project_id}/stream/tags
	Querying a List of Streams with Specified Tags	POST /v1.1/{project_id}/stream/resource_instances/action

3 Environmental Preparations

3.1 Authenticating API Requests

API requests are authenticated using an access key ID/secret access key (AK/SK) file. For more information, see **Authenticating API Requests Using the AK/SK**.

3.2 Authenticating API Requests Using the AK/SK

When you use an API gateway to send requests to underlying services, the requests need to be signed using the AK/SK.

NOTE

AK: is a unique identifier associated with the SK. The AK and SK are used together to sign requests cryptographically

SK: is a key that is used in conjunction with the AK to cryptographically sign requests. Signing a request identifies the sender and prevents the request from being modified.

3.2.1 AK and SK Generation

Before using REST APIs provided by DLI, download the AK/SK certificate to your PC.

NOTE

Each user needs to download the AK/SK certificate only once.

Procedure

Step 1 Log in to the management console.

Step 2 Locate the username in the upper right corner, hover the mouse over it, and select **My Credential** from the drop-down list.

Step 3 Click **Access Credentials**.

Step 4 Click **Add Access Key** to switch to the **Add Access Key** page.

Step 5 Enter the password used for the current login.

Step 6 Enter the authentication code received through email or SMS.

NOTE

- For users created in IAM, if no email address or mobile phone number is specified during user creation, you only need to enter the login password for verification.
- For MyWorkplace users, you do not need to authenticate the login password. If the email address or mobile phone information is unavailable, you can create an access password without the authentication code.

Step 7 Click **OK** to download the access key.

NOTE

To prevent the access key from being leaked, keep it secure.

----End

3.2.2 Request Signing Procedure

Preparations

1. Download the API gateway signing tool from the following website:

   https://apig-demo.obs.eu-de.otc.t-systems.com/java/java-sdk-core.zip
2. Decompress the package.
3. Create a Java project, and reference the extracted JAR to the dependency path.

Signing a Request

1. Create request **com.cloud.sdk.DefaultRequest (JAVA)** that will be signed.
2. Set the target API URL, HTTPS method, and content of request **com.cloud.sdk.DefaultRequest (JAVA)**.
3. Sign request **com.cloud.sdk.DefaultRequest (JAVA)**.
   a. Call **SignerFactory.getSigner(String serviceName, String regionName)** to obtain a sample generated by a signing tool.
   b. Call **Signer.sign(Request<?> request, Credentials credentials)** to sign the request created in **1**.

      The following code shows the details.

      ```
      //Select an algorithm for request signing.
      Signer signer = SignerFactory.getSigner(serviceName, region);
      //Sign the request. The request will change after the signing.
      signer.sign(request, new BasicCredentials(this.ak, this.sk));
      ```
4. Convert the request signed in the previous step to a new request that is suitable to be sent and copy the header of the signed request to the new request.

   For example, if Apache HttpClient is used, convert DefaultRequest to HttpRequestBase and copy the header of the signed DefaultRequest to HttpRequestBase.

3.2.3 Sample Code

The following sample code shows how to sign a request and how to use an HTTP client to send an HTTPS request:

Code is divided into three categories to demonstrate request signing and HTTP request sending.

AccessService: indicates the abstract category that converts the GET, POST, PUT, and DELETE methods into the access method.

Demo: indicates the execution entry used to simulate GET, POST, PUT, and DELETE request sending.

AccessServiceImpl: indicates the implementation of the access method. Code required for API gateway communication is in the access method.

For details about the region and serviceName (service name abbreviation) in the following example code, see .

AccessService.java:

```
package com.cloud.apigateway.sdk.demo;

import java.io.InputStream;
import java.net.URL;
import java.util.Map;

import org.apache.http.HttpResponse;

import com.cloud.sdk.http.HttpMethodName;

public abstract class AccessService {

    protected String serviceName = null;

    protected String region = null;

    protected String ak = null;

    protected String sk = null;

    public AccessService(String serviceName, String region, String ak, String sk)
{
        this.region = region;
        this.serviceName = serviceName;
        this.ak = ak;
        this.sk = sk;
    }

    public abstract HttpResponse access(URL url, Map<String, String> header,
InputStream content, Long contentLength,
        HttpMethodName httpMethod) throws Exception;

    public HttpResponse access(URL url, Map<String, String> header,
HttpMethodName httpMethod) throws Exception {
        return this.access(url, header, null, 0l, httpMethod);
    }

    publicpu  HttpResponse access(URL url, InputStream content, Long
contentLength, HttpMethodName httpMethod)
        throwsth  Exception {
        return this.access(url, null, content, contentLength, httpMethod);
    }
    public HttpResponse access(URL url, HttpMethodName httpMethod) throws
Exception {
        return this.access(url, null, null, 0l, httpMethod);
    }

    public abstract void close();

    public String getServiceName() {
        return serviceName;
    }

    public void setServiceName(String serviceName) {
```

```
        this.serviceName = serviceName;
    }

    public String getRegion() {
        return region;
    }

    public void setRegion(String region) {
        this.region = region;
    }

    public String getAk() {
        return ak;
    }

    public void setAk(String ak) {
        this.ak = ak;
    }

    public String getSk() {
        return sk;
    }

    public void setSk(String sk) {
        this.sk = sk;
    }
}
```

AccessServiceImpl.java:

```
package com.cloud.apigateway.sdk.demo;

import java.io.IOException;
import java.io.InputStream;
import java.net.URISyntaxException;
import java.net.URL;
import java.util.HashMap;
import java.util.Map;

import javax.net.ssl.SSLContext;

import org.apache.http.Header;
import org.apache.http.HttpHeaders;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpDelete;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpHead;
import org.apache.http.client.methods.HttpPatch;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.client.methods.HttpRequestBase;
import org.apache.http.conn.ssl.AllowAllHostnameVerifier;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.conn.ssl.SSLContexts;
import org.apache.http.conn.ssl.TrustSelfSignedStrategy;
import org.apache.http.entity.InputStreamEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;

import com.cloud.sdk.DefaultRequest;
import com.cloud.sdk.Request;
import com.cloud.sdk.auth.credentials.BasicCredentials;
import com.cloud.sdk.auth.signer.Signer;
import com.cloud.sdk.auth.signer.SignerFactory;
import com.cloud.sdk.http.HttpMethodName;


public class AccessServiceImpl extends AccessService {

 private CloseableHttpClient client = null;
```

```
public AccessServiceImpl(String serviceName, String region, String ak,
  String sk) {
 super(serviceName, region, ak, sk);
}

/** {@inheritDoc} */

public HttpResponse access(URL url, Map<String, String> headers,
  InputStream content, Long contentLength, HttpMethodName httpMethod)
  throws Exception {

 // Make a request for signing.
 Request request = new DefaultRequest(this.serviceName);
 try {
  // Set the request address.
  request.setEndpoint(url.toURI());

  String urlString = url.toString();

  String parameters = null;

   if (urlString.contains("?")) {
       parameters = urlString.substring(urlString.indexOf("?") + 1);
       Map parametersmap = new HashMap<String, String>();

       if (null != parameters && !"".equals(parameters)) {
                   String[] parameterarray = parameters.split("&");

                   for (String p : parameterarray) {
                       String key = p.split("=")[0];
                       String value = p.split("=")[1];
                       parametersmap.put(key, value);
                   }
                   request.setParameters(parametersmap);
         }
    }

 } catch (URISyntaxException e) {
  // It is recommended to add logs in this place.
  e.printStackTrace();
 }
 // Set the request method.
 request.setHttpMethod(httpMethod);
 if (headers != null) {
  // Add request header information if required.
  request.setHeaders(headers);
 }
 // Configure the request content.
 request.setContent(content);

 // Select an algorithm for request signing.
 Signer signer = SignerFactory.getSigner(serviceName, region);
 // Sign the request, and the request will change after the signing.
 signer.sign(request, new BasicCredentials(this.ak, this.sk));

 // Make a request that can be sent by the HTTP client.
 HttpRequestBase httpRequestBase = createRequest(url, null,
   request.getContent(), contentLength, httpMethod);
 Map<String, String> requestHeaders = request.getHeaders();
 // Put the header of the signed request to the new request.
 for (String key : requestHeaders.keySet()) {
  if (key.equalsIgnoreCase(HttpHeaders.CONTENT_LENGTH.toString())) {
   continue;
  }
  httpRequestBase.addHeader(key, requestHeaders.get(key));
 }

 HttpResponse response = null;
```

```
 SSLContext sslContext = SSLContexts.custom()
   .loadTrustMaterial(null, new TrustSelfSignedStrategy())
   .useTLS().build();
 SSLConnectionSocketFactory sslSocketFactory = new SSLConnectionSocketFactory(
   sslContext, new AllowAllHostnameVerifier());

 client = HttpClients.custom().setSSLSocketFactory(sslSocketFactory)
   .build();
 // Send the request, and a response will be returned.
 response = client.execute(httpRequestBase);
 return response;
}

/**
 * Make a request that can be sent by the HTTP client.
 *
 * @param url
 *            specifies the API access path.
 * @param header
 *            specifies the header information to be added.
 * @param content
 *            specifies the body content to be sent in the API call.
 * @param contentLength
 *            specifies the length of the content. This parameter is optional.
 * @param httpMethod
 *            specifies the HTTP method to be used.
 * @return specifies the request that can be sent by an HTTP client.
 */
private static HttpRequestBase createRequest(URL url, Header header,
  InputStream content, Long contentLength, HttpMethodName httpMethod) {

 HttpRequestBase httpRequest;
 if (httpMethod == HttpMethodName.POST) {
  HttpPost postMethod = new HttpPost(url.toString());

  if (content != null) {
   InputStreamEntity entity = new InputStreamEntity(content,
     contentLength);
   postMethod.setEntity(entity);
  }
  httpRequest = postMethod;
 } else if (httpMethod == HttpMethodName.PUT) {
  HttpPut putMethod = new HttpPut(url.toString());
  httpRequest = putMethod;

  if (content != null) {
   InputStreamEntity entity = new InputStreamEntity(content,
     contentLength);
   putMethod.setEntity(entity);
  }
 } else if (httpMethod == HttpMethodName.PATCH) {
  HttpPatch patchMethod = new HttpPatch(url.toString());
  httpRequest = patchMethod;

  if (content != null) {
   InputStreamEntity entity = new InputStreamEntity(content,
     contentLength);
   patchMethod.setEntity(entity);
  }
 } else if (httpMethod == HttpMethodName.GET) {
  httpRequest = new HttpGet(url.toString());
 } else if (httpMethod == HttpMethodName.DELETE) {
  httpRequest = new HttpDelete(url.toString());
 } else if (httpMethod == HttpMethodName.HEAD) {
  httpRequest = new HttpHead(url.toString());
 } else {
  throw new RuntimeException("Unknown HTTP method name: "
    + httpMethod);
 }
```

```
  httpRequest.addHeader(header);
  return httpRequest;
 }

 @Override
 public void close() {
  try {
   if (client != null) {
    client.close();
   }
  } catch (IOException e) {
   // It is recommended to add logs in this place.
   e.printStackTrace();
  }
 }
}
```

Demo.java:

```
package com.cloud.apigateway.sdk.demo;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;

import org.apache.http.HttpResponse;

import com.cloud.sdk.http.HttpMethodName;


public class Demo {

 //replace real region
 private static final String region = "regionName";

 //replace real service name
 private static final String serviceName = "serviceName";

 public static void main(String[] args) {

  //replace real AK
  String ak = "akString";
  //replace real SK
  String sk = "skString";

  // get method
  //replace real url
  String url = "urlString";
  get(ak, sk, url);

  // post method
  //replace real url
  String postUrl = "urlString";
  //replace real body
  String postbody = "bodyString";
  post(ak, sk, postUrl, postbody);

  // put method
  //replace real body
  String putbody = "bodyString";
  //replace real url
  String putUrl = "urlString";
  put(ak, sk, putUrl, putbody);

  // delete method
```

```
 //replace real url
 String deleteUrl = "urlString";
 delete(ak, sk, deleteUrl);
}

public static void put(String ak, String sk, String requestUrl,
  String putBody) {

 AccessService accessService = null;
 try {
  accessService = new AccessServiceImpl(serviceName, region, ak, sk);
  URL url = new URL(requestUrl);
  HttpMethodName httpMethod = HttpMethodName.PUT;

  InputStream content = new ByteArrayInputStream(putBody.getBytes());
  HttpResponse response = accessService.access(url, content,
    (long) putBody.getBytes().length, httpMethod);

  System.out.println(response.getStatusLine().getStatusCode());


 } catch (Exception e) {
  e.printStackTrace();
 } finally {
  accessService.close();
 }

}

public static void patch(String ak, String sk, String requestUrl,
  String putBody) {

 AccessService accessService = null;
 try {
  accessService = new AccessServiceImpl(serviceName, region, ak, sk);
  URL url = new URL(requestUrl);
  HttpMethodName httpMethod = HttpMethodName.PATCH;
  InputStream content = new ByteArrayInputStream(putBody.getBytes());
  HttpResponse response = accessService.access(url, content,
    (long) putBody.getBytes().length, httpMethod);

  System.out.println(convertStreamToString(response.getEntity()
    .getContent()));
 } catch (Exception e) {
  e.printStackTrace();
 } finally {
  accessService.close();
 }

}

public static void delete(String ak, String sk, String requestUrl) {

 AccessService accessService = null;

 try {
  accessService = new AccessServiceImpl(serviceName, region, ak, sk);
  URL url = new URL(requestUrl);
  HttpMethodName httpMethod = HttpMethodName.DELETE;

  HttpResponse response = accessService.access(url, httpMethod);
  System.out.println(convertStreamToString(response.getEntity()
    .getContent()));
 } catch (Exception e) {
  e.printStackTrace();
 } finally {
  accessService.close();
 }
```

```
}

public static void get(String ak, String sk, String requestUrl) {

 AccessService accessService = null;

 try {
  accessService = new AccessServiceImpl(serviceName, region, ak, sk);
  URL url = new URL(requestUrl);
  HttpMethodName httpMethod = HttpMethodName.GET;
  HttpResponse response;
  response = accessService.access(url, httpMethod);
  System.out.println(convertStreamToString(response.getEntity()
    .getContent()));
 } catch (Exception e) {
  e.printStackTrace();
 } finally {
  accessService.close();
 }

}

public static void post(String ak, String sk, String requestUrl,
  String postbody) {

 AccessService accessService = new AccessServiceImpl(serviceName,
   region, ak, sk);
 URL url = null;
 try {
  url = new URL(requestUrl);
 } catch (MalformedURLException e) {
  e.printStackTrace();
 }
 InputStream content = new ByteArrayInputStream(postbody.getBytes());
 HttpMethodName httpMethod = HttpMethodName.POST;
 HttpResponse response;

 try {
  response = accessService.access(url, content,
    (long) postbody.getBytes().length, httpMethod);
  System.out.println(convertStreamToString(response.getEntity()
    .getContent()));
 } catch (Exception e) {
  e.printStackTrace();
 } finally {
  accessService.close();
 }
}

private static String convertStreamToString(InputStream is) {
 BufferedReader reader = new BufferedReader(new InputStreamReader(is));
 StringBuilder sb = new StringBuilder();

 String line = null;
 try {
  while ((line = reader.readLine()) != null) {
   sb.append(line + "\n");
  }
 } catch (IOException e) {
  e.printStackTrace();
 } finally {
  try {
   is.close();
  } catch (IOException e) {
   e.printStackTrace();
  }
 }

 return sb.toString();
```

```
  }
}
```

NOTE

1. Parameters **URI**, **AK**, **SK**, and **HTTP METHOD** are mandatory.
2. You can use **request.addHeader()** to add header information.

3.2.4 Sample Code

The following sample code shows how to sign a request and use an HTTP client to send an HTTPS request: Sample code is divided into three categories to demonstrate request signing and HTTP request sending.

- **AccessService**: indicates the abstract category that converts the GET, POST, PUT, and DELETE methods into the access method.
- **Demo**: indicates the execution entry used to simulate GET, POST, PUT, and DELETE request sending.
- **AccessServiceImpl**: indicates the implementation of the **access** method. Code required for API gateway communication is in the access method.

Download sample code package **SdkDemo.zip** from **https://apig-demo.obs.eu-de.otc.t-systems.com/java/SdkDemo.zip**.

Decompress the package to obtain the following files in **SdkDemo\src\main\java\com\cloud\apigateway\sdk\demo**:

- AccessService.java
- Demo.java
- AccessServiceImpl.java

For details about **region** and **serviceName** in the sample code, see **Regions and Endpoints**.

NOTE

1. Parameters **URI**, **AK**, **SK**, and **HTTP METHOD** are mandatory.
2. You can use **request.addHeader()** to add header information.

3.3 Obtaining Project IDs

A project ID is required for some URIs when APIs are called. You can obtain a project ID on the management console

To obtain the project ID, perform the following steps:

1. Register and log in to the DIS management console.
2. Click the username and select **My Credential** from the drop-down list.

   On the displayed page, view the project ID in the project list. For example, project_id:"5a3314075bfa49b9ae360f4ecd333695".

4 API Usage

4.1 REST API Overview

DIS provides REST APIs.

In REST, specific information or data on a network is represented by a resource, which is referenced with a uniform resource identifier (URI). Clients on a network can locate resources using uniform resource locators (URLs).

The format of a URL is as follows: https://***Endpoint*/*uri***

Table 4-1 describes URL parameters.

Table 4-1 Parameter description

Parameter	Description
Endpoint	Name of the server that provides the requested resource. For more information about regions and endpoints, see **Regions and Endpoints**.
uri	Path in which the requested resource is located. You can obtain the path from the URI of a specific API.

A REST API request or response consists of the following five parts:

- Request URI
- Request header
- Request body
- Response header
- Response body

Request URL

The format of a request URL is as follows;

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

Table 4-2 Parameter description

Parameter	Description
URL-scheme	Specifies the protocol used to transfer requests. The DIS API must use HTTPS.
Endpoint	Specifies the domain name or IP address of the server bearing the REST service endpoint. Obtain this value from **Regions and Endpoints**.
resource-path	Specifies the resource path, that is, the API access path. You can obtain the path from the URI of a specific API.
query-string	(Optional) For example, API version or resource selection criteria.

Request Header

A request header consists of an HTTP method and additional field in the request header.

- HTTP method, also known as an operation or action, used to specify how to access a specific resource.

  Table 4-3 describes the HTTP methods supported by the REST API of DIS.

  Table 4-3 HTTP methods

Method	Description
GET	Requests the server to return a specific resource.
POST	Requests the server to add a resource or perform special operations.
PUT	Requests the server to update a specific resource.
DELETE	Requests the server to delete a specific resource, for example, to delete an object.
HEAD	Requests the server to obtain the resource header.
PATCH	Requests the server to update the partial content of a specific resource. If the resource does not exist, a resource can be created using the PATCH method.

- (Optional) Additional field in the request header required by a specific URI and HTTP method

  For details about common request headers, see **Table 4-4**. For details about the request authentication information, see **Authenticating API Requests**.

Table 4-4 Common request message headers

Header	Mandatory	Type	Description
Content-Type	Yes	String	Type of the resource content. For example: application/json Default value: none
X-Sdk-Date	Yes	String	Time to send a request. The time format is *YYYYMMDD*'**T**'*HHMMSS*'**Z**'. For example: 20180820T101459Z Default value: none
Authorization	Yes	String	Authorization information in a request. SDK-HMAC-SHA256 Credential=ZIRRKMTWPTQFQI1WKNKB/20150907/eu-de/ec2/sdk_request, SignedHeaders=content-type;host;x-sdk-date, Signature=55741b610f3c9fa3ae40b5a8021ebf7ebc2a28a603fc62d25cb3bfe6608e1994 Default value: none
Host	Yes	String	Requested service information. Default value: none

Request Body

A request body is encapsulated and sent in the JSON format, and is used to transfer content other than the request header. The format is specified by **Content-Type** in the request header.

Which fields are mandatory and optional in an HTTP request varies with different URI objects.

Response Header

A response header consists of an HTTP status code and additional field in the response header:

- HTTP status code. For details, see **Status Codes**.
- Additional field in the response header required by a specific response, for example, the **Content-Type** response header.

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

Table 4-5 Common response message headers

Header	Type	Description
Content-Type	String	Type of the resource content. For example: application/json Default value: none
Date	String	DIS system response time. For example: Mon, 15 Jul 2013 21:08:05 GMT Default value: none

Response Body

The response body is encapsulated as JSON-formatted text, and is used to transfer content other than the response header. The format is specified by **Content-Type** in the response header.

Initiating a Request

There are three methods to initiate a request to the server end:

- cURL

  cURL is a command-line tool used to perform URL operations and transmit information. It can serve as an HTTP client to send HTTP requests to the server end and receive response messages. cURL is suitable for use in API tuning scenarios. For more information about cURL, visit **https://curl.haxx.se/**.
- Code

  You can call an API through code to assemble, send, and process requests.
- REST client

  Mozilla Firefox and Google Chrome provide a graphical browser plug-in for REST clients to send and process requests.

  For Mozilla Firefox, see **Firefox REST Client**.

  For Google Chrome, see **Postman Interceptor**.

4.2 API Calling Process

The following describes the procedure of calling a DIS API.

1. Before you call an API, obtain the request authentication information, and fill it in the request header. For details, see **Authenticating API Requests Using the AK/SK**.
2. Constructing a request

   Configure the request parameters to construct a request. For details, see **Request URL**, **Request Header**, and **Request Body**.
3. Initiating a request

   For details, see **Initiating a Request**.
4. Parsing a response

For details, see **Request URL**, **Response Header**, and **Response Body**.

5 API Description

5.1 Creating a DIS Stream

Function

This API is used to create a DIS stream.

URL

- URL format

  POST /v2/{project_id}/streams
- Parameter description

  None

Request

- Example request
  - Create a stream whose source data type is BLOB. Dump tasks cannot be created for such a stream.

```
POST https://{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/
streams
Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
Authorization: SDK-HMAC-SHA256
Credential=QRUP2R3QFNAOVAWMYHZW/20180820/
eu-de/test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480c
e
{
    "stream_name": "dis-DLpR",
    "partition_count": 1,
    "stream_type": "COMMON",
    "data_type": "BLOB",
    "data_duration": 24
}
```

  - Create a stream whose source data type is JSON or CSV. Dump tasks cannot be created for such a stream.

```
POST https://{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/
streams
Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
Authorization: SDK-HMAC-SHA256
Credential=QRUP2R3QFNAOVAWMYHZW/20180820/
eu-de/test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480c
e
{
    "stream_name": "dis-DLpR",
    "partition_count": 1,
    "stream_type": "COMMON",
        "data_type": "JSON",
    "data_duration": 24,
        "data_schema":"{\"type\":\"record\",\"name\":\"RecordName\",
\"fields\":[{\"name\":\"id\",\"type\":\"string\",\"doc\":\"Type inferred
from '\\\"2017/10/11 11:11:11\\\"'\"},{\"name\":\"info\",\"type\":{\"type
\":\"array\",\"items\":{\"type\":\"record\",\"name\":\"info\",\"fields\":
[{\"name\":\"date\",\"type\":\"string\",\"doc\":\"Type inferred from '\\
\"2018/10/11 11:11:11\\\"'\"}]}},\"doc\":\"Type inferred from '[{\\\"date
\\\":\\\"2018/10/11 11:11:11\\\"}]'\"}]}"}
}
```

- Parameter description

Table 5-1 Parameter description

Parameter	Mandatory	Type	Description
stream_name	Yes	String	Name of the DIS stream. Each DIS stream has a unique name. A stream name is 1 to 64 characters in length. Only letters, digits, hyphens (-), and underscores (_) are allowed.
stream_type	No	String	Stream type. Possible values: - **Common**: indicates a common stream. The bandwidth is 1 MB/s. - **ADVANCED**: indicates an advanced stream. The bandwidth is 5 MB/s. Default value: COMMON

Parameter	Mandatory	Type	Description
partition_count	Yes	Int	Quantity of the partitions into which data records in the newly created DIS stream will be distributed. Partitions are the base throughput unit of a DIS stream. The value range varies depending on the value of **stream_type**. ● If **stream_type** is not specified or set to **COMMON**, the value of **partition_count** is an integer from 1 to 50. If the tenant has created **N** common partitions, the maximum value of **partition_count** is **50-N**. ● If **stream_type** is set to **ADVANCED**, the value of **partition_count** is an integer from 1 to 10. If the tenant has created **N** advanced partitions, the maximum value of **partition_count** is **10-N**.
data_duration	No	Int	Period of time for which data is retained in the DIS stream. Value range: *N* x 24, where *N* is an integer from 1 to 7. Unit: hour Default value: 24 If this parameter is left unspecified, the default value will be used.
data_schema	No	String	Source data structure that defines JOSN and CSV formats. It is described in the syntax of Avro. For details about Avro, see **http://avro.apache.org/docs/current/#schemas**. **NOTE** This parameter is mandatory when **Dump File Format** is **parquet** or **carbon**.
tags	No	List<Tag>	Label of the stream.

Parameter	Mandatory	Type	Description
obs_destination_descriptor	No	Object	Parameter list of the OBS to which data in the DIS stream will be dumped. Data in a DIS stream cannot be dumped to multiple destinations. By default, this parameter is left unspecified, indicating that data in the DIS stream will not be dumped to OBS. **NOTE** This parameter is available and needs to be configured only when **Source Data Type** is set to **FILE**.

Table 5-2 obs_destination_descriptor parameter description

Parameter	Mandatory	Type	Description
agency_name	Yes	String	Name of the agency created in IAM. DIS uses an agency to access your specified resources. IAM agency parameter settings: ● **Agency Type**: **Cloud service** ● **Cloud Service**: **DIS** ● **Validity Period**: **Permanent** ● Set **Policy** to **Tenant Administrator** on the OBS project in the **Global** service region. This parameter cannot be left unspecified and the parameter value cannot exceed 64 characters.
obs_bucket_path	Yes	String	Name of the OBS bucket used to store data from the DIS stream.

Table 5-3 Tag parameter description

Parameter	Mandatory	Type	Description
key	Yes	String	Key. A tag key cannot contain special characters such as =*<>\,|/ or start or end with a space.
value	Yes	String	Value. A tag value cannot contain special characters such as =*<>\,|/ or start or end with a space.

Response

- If the DIS stream was successfully created, a 201 response with an empty response body is returned.
- If the DIS stream failed to be created, identify the failure cause according to the response body and the instructions in **Error Codes**.

Status Code

- Normal

  201 Created
- Failed

  For more information, see **Error Codes**.

5.2 Deleting a DIS Stream

Function

This API is used to delete a DIS stream after pushing or pulling data to or from the stream or if the stream does not run properly.

URL

- URL format

  DELETE /v2/{project_id}/streams/{stream_name}
- Parameter description

  None

Request

- Example request

```
DELETE https://{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/streams/
stream_test

Request Header:
Content-Type: application/json
Authorization: SDK-HMAC-SHA256
Credential=QRUP2R3QFNAOVAWMYHZW/20180820/eu-de/test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480ce
```

- Parameter description

  None

Response

- If the DIS stream was successfully deleted, a 204 response with an empty response body is returned. For details about status codes, see **Status Codes**.
- If the DIS stream failed to be created, identify the failure cause according to the response body and the instructions in **Error Codes**.

Status Code

- Normal

  204 NOCONTENT
- Failed

  For more information, see **Error Codes**.

5.3 Listing DIS Streams

Function

This API is used to list all the DIS streams created by the current tenant.

URL

- URL format

  GET /v2/{project_id}/streams
- Parameter description

  None

Request

- Example request

```
GET https://{ip}{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/streams?
limit=10&start_stream_name=test_stream

Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
Authorization: SDK-HMAC-SHA256 Credential=QRUP2R3QFNAOVAWMYHZW/20180820/eu-de/
test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480ce
```

- Parameter description

Table 5-4 Parameter description

Parameter	Mandatory	Type	Description
limit	No	Int	The maximum number of DIS streams to list in a single API call. Value range: 1 to 100 Default value: 10
start_stream_name	No	String	Name of the DIS stream to start the list with. The returned stream list does not contain this DIS stream name. **NOTE** Streams in the stream list are arranged in alphabetical order.

Response

- Example response

```
{
  "total_number": 1,
  "has_more_streams": false,
  "stream_names": [
    "stream_test"
  ]
}
```

- Parameter description

Table 5-5 Response parameter description

Parameter	Type	Description
total_number	Int	Total number of DIS streams created by the current tenant.
stream_names	List<String>	Name of the list of the streams meeting the current requests.
has_more_streams	Boolean	Specify whether there are more matching DIS streams to list. Possible values: ● **true**: There are more streams. ● **false**: There are no more streams.

Status Code

- Normal

  200 OK
- Failed

  For more information, see **Error Codes**.

5.4 Viewing Details of a DIS Stream

Function

This API is used to view details about a specified DIS stream.

URL

- URL format

  GET /v2/{project_id}/streams/{stream_name}
- Parameter description

  None

Request

- Example request

```
GGET https://{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/streams/
stream_test?start_partitionId=shardId-0000000000&limit_partitions=100
```

```
Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
Authorization: SDK-HMAC-SHA256 Credential=QRUP2R3QFNAOVAWMYHZW/20180820/eu-de/
test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480ce
```

- Parameter description

Table 5-6 Parameter description

Parameter	Mandatory	Type	Description
stream_name	Yes	String	Name of the DIS stream to be queried.
start_partitionId	No	String	ID of the partition to start the partition list with. The returned partition list does not contain this partition ID.
limit_partitions	No	Int	Maximum number of partitions to list in a single API call. Value range: 1 – 1000 Default value: 100

Response

- Example response

```
{
  "stream_name": "stream_test",
  "stream_id": "U7v0U582F8ccXyErJKC",
  "create_time": 1504679587519,
  "last_modified_time": 1504679587519,
  "retention_period": 24,
  "status": "RUNNING",
  "stream_type": "COMMON",
  "data_type": "JSON",
  "writable_partition_count": 3,
  "readable_partition_count": 5,
  "partitions": [
    {
     "status": "ACTIVE",
     "partition_id": "shardId-0000000000",
      "hash_range": "[0 : 9223372036854775807]",
       "sequence_number_range": "[0 : 200]"
    }
],
"data_schema":{
   "type": "record",
   "name": "RecordName",
   "fields": [{
       "name": "id",
       "type": "string",
       "doc": "Type inferred from '\"1\"'"
               }, {
       "name": "detail",
       "type": {
           "type": "record",
           "name": "detail",
           "fields": [{
               "name": "detID",
```

```
                "type": "string",
                "doc": "Type inferred from '\"05790110000000000103\"'"
            }, {
                "name": "endTime",
                "type": "string",
                "doc": "Type inferred from '\"2018/10/07 13:26:35\"'"
            }]
        },
        "doc": "Type inferred from '{\"detID\":\"05790110000000000103\",
\"endTime\":\"2018/10/07 13:26:35\"}'"
    }]
   },
  "has_more_partitions": false,
   "tags": []
}
```

- Parameter description

Table 5-7 Response parameter description

Parameter	Type	Description
stream_name	String	Name of the DIS stream.
stream_id	String	Unique ID of each stream.
create_time	Long	Timestamp at which the DIS stream was created.
last_modified_time	Long	Timestamp at which the DIS stream was most recently modified.
retention_period	Int	Period of time for which data is retained in the DIS stream.
status	String	The stream status is one of the following:
stream_type	String	Partition type.
data_type	String	Type of the source data.
data_schema	String	Source data structure that defines JOSN and CSV formats. It is described in the syntax of Avro. For details about Avro, see **http://avro.apache.org/docs/current/#schemas**.
writable_partition_count	Int	Total number of writable partitions (including partitions in ACTIVE and DELETED states).
readable_partition_count	Int	Total number of readable partitions (including partitions in ACTIVE state only).
tags	List<Tag>	Label of the stream.
partitions	List<PartitionResult>	A list of partitions that comprise the DIS stream. For more information, see **Table 5-8**.
has_more_partitions	Boolean	Specify whether there are more matching partitions of the DIS stream to list. ● **true**: There are more partitions. ● **false**: There are no more partitions.

Table 5-8 PartitionResult parameter description

Parameter	Type	Description
status	String	Current status of each partition. ● CREATING ● ACTIVE ● DELETED ● EXPIRED
partition_id	String	Unique ID of the partition.
hash_range	String	Possible value range of the hash key used by each partition.
sequence_number_range	String	Sequence number range of each partition.

Table 5-9 Tag parameter description

Parameter	Mandatory	Type	Description
key	Yes	String	Key. A tag key cannot contain special characters such as =*<>\,|/ or start or end with a space.
value	Yes	String	Value. A tag value cannot contain special characters such as =*<>\,|/ or start or end with a space.

Status Code

- Normal
  200 OK
- Failed
  For more information, see **Error Codes**.

5.5 Putting Data into a DIS Stream

Function

This API is used to put data into DIS streams.

URL

- URL format
  POST /v2/{project_id}/records
- Parameter description

None

Request

- Example request

```
POST https://{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/records

Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
Authorization: SDK-HMAC-SHA256 Credential=QRUP2R3QFNAOVAWMYHZW/20180820/eu-de/
test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480ce


{
 "stream_name": "stream_test",
  "records": [
    {
      "data":
"MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTE=",
      "explicit_hash_key": null,
      "partition_key": "0"
 ]
}
```

- Parameter description

Table 5-10 Parameter description

Parameter	Mandatory	Type	Description
stream_name	Yes	String	Name of the stream created on the management console. A stream name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.
records	Yes	List<PutRecordsRequestEntry>	Information of data records. For more information, see **Table 5-11**.

Table 5-11 records parameter description

Parameter	Mandatory	Type	Description
data	Yes	Base64-encoded binary data object	Data to be put into the chosen DIS stream. The uploaded data must be serialized into binary Base64-encoded data.

Parameter	Mandatory	Type	Description
explicit_hash_key	No	String	Hash value used to explicitly determine the partition into which an individual data record will be put. The hash value overrides the partition key hash.
partition_id	No	String	Unique ID of the partition.
partition_key	No	String	Partition key of the partition into which an individual data record will be put. **NOTE** The **partition_id** parameter takes precedence over the **partition_key** parameter. If the **partition_id** parameter is not passed in, then the **partition_key** parameter is selected for use.

NOTE

- The user's raw data may contain invisible characters. Therefore, the raw data must be encoded using Base64 encoding before invoking the API.
- If a user uploads data using SDK provided by DIS, the SDK will automatically encode the raw data using Base64 encoding.

Response

- Example response

```
{
   "failed_record_count": 0,
   "records": [
      {


         "sequence_number": "195",
         "partition_id": "shardId-0000000000"
      }
   ]
}
```

- Parameter description

Table 5-12 Response parameter description

Parameter	Type	Description
failed_record_count	Int	The number of data records that failed to be put into the selected DIS stream.
records	List<PutRecordResult>	Processing result of each data record. For more information, see **Table 5-13**.

Table 5-13 PutRecordResult parameter description

Parameter	Type	Description
error_code	String	Error code for an individual record result.
error_message	String	Error message for an individual record result.
partition_id	String	Partition ID for an individual record result.
sequence_number	String	Sequence number of an individual data record. Each data record has a sequence number that is unique within its partition. The sequence number is assigned by DIS when a data producer calls PutRecords to add data to a DIS stream. Sequence numbers for the same partition key generally increase over time; the longer the time period between write requests (PutRecords requests), the larger the sequence numbers become.

Status Code

- Normal

  200 OK
- Failed

  For more information, see **Error Codes**.

5.6 Pulling Data from a DIS Stream

Function

This API is used to pull data from a DIS stream.

URL

- URL format

  GET /v2/{project_id}/records{?partition-cursor}
- Parameter description

  None

Request

- Example request

```
GET https://{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/records?
partition-cursor=
eyJpdGVyR2VuVGltZSI6MTQ5MDk0ODk5OTM5NywiU3RyZWFtTmFtZSI6IjY2MCIsIlNoYXJkSWQiOi
IwIiwiU2hhcmRJdGVyYXRvclR5cGUiOiJBVF9TRVFVRU5DRV9OVU1CRVIiLCJTdGFydGluZ1NlcXVl
bmNlTnVtYmVyIjoiMCIsIlRpbWVTdGFtcCI6MH0=

Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
```

```
Authorization: SDK-HMAC-SHA256 Credential=QRUP2R3QFNAOVAWMYHZW/20180820/eu-de/
test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480ce
```

- Parameter description

Table 5-14 Parameter description

Parameter	Mandatory	Type	Description
partition-cursor	Yes	String	Cursor of the partition used to specify the position in the partition from which to start reading data records sequentially. Value: 1 to 512 characters **NOTE** The validity period of a cursor is 5 minutes.

Response

- Example response

```
{
  "records": [
    {
      "partition_key": "0",
      "sequence_number": "21",
      "data":
      "timestamp": 1527577402541,
      "timestamp_type": "CreateTime"
    }
  ],
  "next_partition_cursor":
"eyJpdGVyR2VuVGltZSI6MTQ5MDk1MDE1Nzc0NywiU3RyZWFtTmFtZSI6IjY2MCIsIlNoYXJkSWQiO
iIwIiwiU2hhcmRJdGVyYXRvclR5cGUiOiJBVF9TRVFVRU5DRV9OVU1CRVIiLCJTdGFydGluZ1NlcXV
lbmNlTnVtYmVyIjoiMjIiLCJUaW1lU3RhbXAiOjB9",
 }
```

- Parameter description

Table 5-15 Response parameter description

Parameter	Type	Description
records	List<PullRecord>	Information of data records. For more information, see **Table 5-16**.
next_partition_cursor	String	Next cursor, which specifies the next position in the partition from which to start reading data records sequentially. **NOTE** The validity period of a cursor is 5 minutes.

Table 5-16 PullRecord parameter description

Parameter	Type	Description
data	String	Data pulled from the DIS stream.
partition_key	String	Partition key set when data is being uploaded. **NOTE** If the **partition_key** parameter is transferred when data is uploaded, this parameter is returned when data is downloaded. If the **partition_key** parameter is not transferred but the **partition_id** parameter is transferred when data is uploaded, the **partition_id** parameter is not returned when data is downloaded.
sequence_number	String	Sequence number of an individual data record. Each data record has a sequence number that is unique within its partition. The sequence number is assigned by DIS when a data producer calls PutRecords to add data to a DIS stream. Sequence numbers for the same partition key generally increase over time; the longer the time period between write requests (PutRecords requests), the larger the sequence numbers become.
timestamp	Long	Timestamp when the record is written to DIS.
timestamp_type	String	Type of the timestamp. The value is **CreateTime**, specifying the creation time.

NOTE

- The data obtained through this API is the data encoded from the raw data using Base64 encoding.
- If a user downloads data using SDK provided by DIS, the SDK will automatically decode the data using Base64 decoding, and the user will obtain the raw data.

Status Code

- Normal

  200 OK
- Failed

  For more information, see **Error Codes**.

5.7 Obtaining a Cursor

Function

This API is used to obtain a cursor. A cursor specifies the partition position from which to start reading data records sequentially.

URL

- URL format

  GET /v2/{project_id}/cursors{?stream-name,partition-id,cursor-type,starting-sequence-number}

- Parameter description

  None

Request

- Example request

```
GET https://{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/cursors?
stream-name=lzc08&partition-id=0&cursor-type=AT_SEQUENCE_NUMBER&starting-
sequence-number=11
Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
Authorization: SDK-HMAC-SHA256
Credential=QRUP2R3QFNAOVAWMYHZW/20180820/eu-de/test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480ce
```

- Parameter description

Table 5-17 Parameter description

Parameter	Mandatory	Type	Description
stream-name	Yes	String	Name of the stream created on the management console. Value range: Only letters, digits, underscores (_), and hyphens (-) are allowed. Only letters, digits, hyphens (-), and underscores (_) are allowed.
partition-id	Yes	String	ID of the partition to get the cursor for. Value range: 0 to 2147483647

Parameter	Mandatory	Type	Description
cursor-type	No	String	Cursor type. ● **AT_SEQUENCE_NUMBER**: The consumer application starts reading from the position denoted by a specific sequence number. This is the default cursor type. ● **AFTER_SEQUENCE_NUMBER**: The consumer application starts reading right after the position denoted by a specific sequence number. ● **TRIM_HORIZON**: The consumer application starts reading at the last untrimmed record in the partition in the system, which is the oldest data record in the partition. ● **LATEST**: The consumer application start reading right after the most recent record in the partition. ● **AT_TIMESTAMP**: The consumer application starts reading from a specific timestamp.
starting-sequence-number	No	String	Sequence number of an individual data record. Each data record has a sequence number that is unique within its partition. The sequence number is assigned by DIS when a data producer calls PutRecords to add data to a DIS stream. Sequence numbers for the same partition key generally increase over time; the longer the time period between write requests (PutRecords requests), the larger the sequence numbers become. Value range: 0 to 9223372036854775807

Parameter	Mandatory	Type	Description
timestamp	No	Long	Timestamp when the data record starts to be read. This parameter is used together with **AT_TIMESTAMP**. **NOTE** ● If the **starting-sequence-number** parameter is transferred, it is used preferentially. If the **starting-sequence-number** parameter is not transferred, the **timestamp** parameter will be used. ● This timestamp is accurate to milliseconds.

Response

- Example response

```
{
 "partition_cursor":
"eyJnZXRJdGVyYXRvclBhcmFtIjp7InN0cmVhbS1uYW1lIjoianpjIiwicGFydGl0aW9uLWlkIjoiMCIsImN1cnNvci10eXBlIjoiQVRfU0VRVUVOQ0VfTlVNQkVSIiwic3RhcnRpbmctc2VxdWVuY2UtbnVtYmVyIjoiMTAifSwiZ2VuZXJhdGVUaW1lc3RhbXAiOjE1MDYxNTk1NjM0MDV9"
}
```

- Parameter description

Table 5-18 Response parameter description

Parameter	Type	Description
partition_cursor	String	Cursor of the partition used to specify the position in the partition from which to start reading data records sequentially. Value: 1 to 512 characters **NOTE** The validity period of a cursor is 5 minutes.

Status Code

- Normal

  200 OK
- Failed

  For more information, see **Error Codes**.

5.8 Creating a Consumer Application

Function

This API is used to create a consumer application.

URL

- URL format

  POST /v2/{project_id}/apps

- Parameter description

  None

Request

- Example request

```
POST https://{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/apps

Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
Authorization: SDK-HMAC-SHA256 Credential=QRUP2R3QFNAOVAWMYHZW/20180820/eu-de/
test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480ce

{
  "app_name": "app_test"
}
```

- Parameter description

Table 5-19 Parameter description

Parameter	Mandatory	Type	Description
app_name	Yes	String	Unique name of the consumer application to be created. An application name is 1 to 50 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.

Response

- If the DIS stream was successfully created, a 201 response with an empty response body is returned.
- If the DIS stream failed to be created, identify the failure cause according to the response body and the instructions in **Error Codes**.

Status Code

- Normal

  201 Created
- Failed

  For more information, see **Error Codes**.

5.9 Deleting a Consumer Application

Function

This API is used to delete a consumer application.

URL

- URL format

  DELETE /v2/{project_id}/apps/{app_name}
- Parameter description

  None

Request

- Example request

```
DELETE
https://{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/apps/apptest

Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
Authorization: SDK-HMAC-SHA256
Credential=QRUP2R3QFNAOVAWMYHZW/20180820/eu-de/test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480ce
```

- Parameter description

  None

Response

- If the DIS stream was successfully deleted, a 204 response with an empty response body is returned. For details about status codes, see **Status Codes**.
- If the DIS stream failed to be created, identify the failure cause according to the response body and the instructions in **Error Codes**.

Status Code

- Normal

  204 NOCONTENT
- Failed

  For more information, see **Error Codes**.

5.10 Adding a Checkpoint

Function

This API is used to add a checkpoint.

URL

- URL format

  POST /v2/{project_id}/checkpoints
- Parameter description

  None

Request

- Example request

```
POST https://{ip}{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/
checkpoints

Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
Authorization: SDK-HMAC-SHA256 Credential=QRUP2R3QFNAOVAWMYHZW/20180820/eu-de/
test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480ce

{
 "stream_name": "stream_name_test1",
  "app_name": "app_name1",
  "partition_id": "shardId-0000000000",
  "sequence_number": "10",
  "metadata": "metadata",
  "checkpoint_type": "LAST_READ"
}
```

- Parameter description

Table 5-20 Parameter description

Parameter	Mandatory	Type	Description
stream_name	Yes	String	Name of the DIS stream whose data record will have a checkpoint. A stream name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.

Parameter	Mandatory	Type	Description
app_name	Yes	String	Unique name of the consumer application that will read data from the chosen DIS stream. An application name must contain 1 to 50 characters. Only letters, digits, hyphens (-), and underscores (_) are allowed.
partition_id	Yes	String	Unique ID of the partition.
sequence_number	Yes	String	Unique sequence number of the record for which a checkpoint will be added.
metadata	No	String	Metadata of the consumer application. The maximum metadata length is 1000 characters.
checkpoint_type	Yes	String	Type of the checkpoint. The checkpoint type **LAST_READ** indicates that only sequence numbers are recorded into the database.

Response

- If the DIS stream was successfully created, a 201 response with an empty response body is returned.
- If the DIS stream failed to be created, identify the failure cause according to the response body and the instructions in **Error Codes**.

Status Code

- Normal

  201 Created
- Failed

  For more information, see **Error Codes**.

5.11 Querying a Checkpoint

Function

This API is used to query a checkpoint.

URL

- URL format

  GET /v2/{project_id}/checkpoints{?stream_name,partition_id,app_name,checkpoint_type}

- Parameter description

  None

Request

- Example request

```
GET https://{ip}{endpoint}:{port}/v2/6352bba7aaab443aa1d9943efc586a68/
checkpoints?
stream_name=stream_name_test&partition_id=0&app_name=app_name&checkpoint_type=
LAST_READ

Request Header:
Content-Type: application/json
X-Sdk-Date: 20180820T024248Z
Authorization: SDK-HMAC-SHA256 Credential=QRUP2R3QFNAOVAWMYHZW/20180820/eu-de/
test/sdk_request,
SignedHeaders=host;x-sdk-date,
Signature=9d8b56b055c0e1f7a9498d881a7cb726be91b4f0cde1773b0b1557e987a480ce
```

- Parameter description

Table 5-21 Parameter description

Parameter	Mandatory	Type	Description
stream_name	Yes	String	Name of the stream created on the console. A stream name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.
partition_id	Yes	String	Unique ID of the partition.
app_name	Yes	String	Unique ID of the consumer application. An application name is 1 to 50 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.
checkpoint_type	Yes	String	Type of the checkpoint. The checkpoint type **LAST_READ** indicates that only sequence numbers are recorded into the database.

Response

- Example response

```
{
  "sequence_number": "10",
  "metadata": "metadata"
}
```

- Parameter description

Table 5-22 Response parameter description

Parameter	Type	Description
sequence_number	String	Unique sequence number. Each data record has a sequence number that is unique within its partition.
metadata	String	Metadata of the consumer application.

NOTE

If the checkpoint does not exist or expires, the value of **sequence_number** is **-1** and the value of **metadata** is empty.

Status Code

- Normal

  200 OK
- Failed

  For more information, see **Error Codes**.

5.12 Tag Management APIs

5.12.1 Adding a Tag to a Specified Stream

Function

This API is used to add a tag to a specified stream.

A stream has a maximum of 10 tags. This API is idempotent. If a tag to be created has the same key as an existing tag, the tag will overwrite the existing one.

URI

POST /v2/{project_id}/stream/{stream_id}/tags

The following table describes URI parameters.

Table 5-23 URI parameter description

Parameter	Mandatory	Description
project_id	Yes	Project ID. For details about how to obtain the project ID, see **Obtaining Project IDs**.
stream_id	Yes	Stream ID.

Request

Request parameters

Table 5-24 Request parameters

Parameter	Mandatory	Type	Description
key	Yes	String	Key. A tag key consists of a maximum of 36 characters, including A-Z, a-z, 0-9, hyphens (-), and underscores (_).
value	Yes	String	Value. The value consists of a maximum of 43 characters, including A-Z, a-z, 0-9, periods (.), hyphens (-), and underscores (_), and can be an empty string.

Response

Response parameter

None

Example

- Request example

```
{
    "tag":
        {
            "key":"DEV",
            "value":"DEV1"
        }
}
```

- Response example

  None

Status Code

Table 5-25 lists the status code of this API.

Table 5-25 Status code

Status Code	Description
204	No Content

5.12.2 Deleting a Tag of a Specified Stream

Function

This API is used to delete a tag of a specified stream.

URI

DELETE /v2/{project_id}/stream/{stream_id}/tags/{key}

The following table describes URI parameters.

Table 5-26 URI parameter description

Parameter	Mandatory	Description
project_id	Yes	Project ID. For details about how to obtain the project ID, see **Obtaining Project IDs**.
stream_id	Yes	Stream ID.

Request

Request parameters

None

Response

Response parameter

None

Example

None

Status Code

Table 5-27 lists the status code of this API.

Table 5-27 Status code

Status Code	Description
204	No Content

5.12.3 Querying a Tag of a Specified Stream

Function

This API is used to query a tag of a specified stream.

URI

GET /v2/{project_id}/stream/{stream_id}/tags

The following table describes URI parameters.

Table 5-28 URI parameter description

Parameter	Mandatory	Description
project_id	Yes	Project ID. For details about how to obtain the project ID, see **Obtaining Project IDs**.
stream_id	Yes	Stream ID.

Request

Request parameters

None

Response

Response parameter

The following table describes the response parameters.

Table 5-29 Response parameters

Parameter	Mandatory	Type	Description
tags	No	List<tag>	Tag list.
key	Yes	String	Key. A tag key consists of a maximum of 36 characters, including A-Z, a-z, 0-9, hyphens (-), and underscores (_).
value	Yes	String	Value. The value consists of a maximum of 43 characters, including A-Z, a-z, 0-9, periods (.), hyphens (-), and underscores (_), and can be an empty string.

Example

- Request example

  None

- Response example

```
{
       "tags": [
        {
            "key": "key1",
            "value": "value1"
        },
        {
            "key": "key2",
            "value": "value3"
        }
    ]
}
```

Status Code

Table 5-30 lists the status code of this API.

Table 5-30 Status code

Status Code	Description
200	OK

5.12.4 Adding or Deleting Stream Tags in Batches

Function

This API is used to add or delete tags to or from a specified stream in batches.

You can add a maximum of 10 tags to a stream.

This API is idempotent.

If a tag to be created has the same key as an existing tag in a stream, the tag will overwrite the existing one.

When tags are being deleted and some tags do not exist, the operation is considered successful by default. The character set of the tags will not be checked. A key and a value can respectively contain up to 36 and 43 Unicode characters. When you delete tags, the tag structure cannot be missing, and the key cannot be left blank or be an empty string.

URI

POST /v2/{project_id}/stream/{stream_id}/tags/action

The following table describes URI parameters.

Table 5-31 URI parameter description

Parameter	Mandatory	Description
project_id	Yes	Project ID. For details about how to obtain the project ID, see **Obtaining Project IDs**.
stream_id	Yes	Stream ID.
tags	Yes	List<resource_tag> List of tags.
action	Yes	String Operation to be performed. The value can be set to **create** or **delete** only.

Request

Request parameters

The following table describes the request parameters.

Table 5-32 Request parameter description

Parameter	Mandatory	Type	Description
key	Yes	String	Key. A tag key consists of a maximum of 36 characters, including A-Z, a-z, 0-9, hyphens (-), and underscores (_).
value	Yes	String	Value. The value consists of a maximum of 43 characters, including A-Z, a-z, 0-9, periods (.), hyphens (-), and underscores (_), and can be an empty string.

Response

Response parameter

None

Example

- Request example

```
{
    "action": "create",
    "tags": [
        {
```

```
            "key": "key1",
            "value": "value1"
        },
        {
            "key": "key",
            "value": "value3"
        }
    ]
}
```

- Response example

  None

Status Code

Table 5-33 lists the status code of this API.

Table 5-33 Status code

Status Code	Description
204	No Content

5.12.5 Querying All Tags

Function

This API is used to query all tag sets of a specified region.

URI

GET /v2/{project_id}/stream/tags

Table 5-34 URI parameter description

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

Request

Request parameters

None

Response

Response parameter

The following table describes the response parameters.

Table 5-35 Response parameters

Parameter	Mandatory	Type	Description
tags	No	List<tag>	Tag list.
key	Yes	String	Key. A tag key consists of a maximum of 36 characters, including A-Z, a-z, 0-9, hyphens (-), and underscores (_).
value	Yes	String	Value. The value consists of a maximum of 43 characters, including A-Z, a-z, 0-9, periods (.), hyphens (-), and underscores (_), and can be an empty string.

Example

- Request example

None

- Response example

```
{
    "tags": [
    {
        "key": "key1",
        "values": [
            "value1",
            "value2"
        ]
        },
        {
        "key": "key2",
        "values": [
            "value1",
            "value2"
        ]
        }
    ]
}
```

Status Code

Table 5-36 lists the status code of this API.

Table 5-36 Status code

Status Code	Description
200	OK

5.12.6 Querying a List of Streams with Specified Tags

Function

This API is used to filter streams by tag.

By default, streams and tags are sorted in descending order of creation time.

URI

POST /v2/{project_id}/stream/resource_instances/action

The following table describes URI parameters.

Table 5-37 URI parameter description

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

Request

Request parameters

Table 5-38 Parameter description

Parameter	Mandatory	Type	Description
tags	No	List<tag>	The return result contains resources corresponding to all tags in this parameter. This parameter contains a maximum of 10 keys, and each key contains a maximum of 10 values. The structure body cannot be missing, and the key cannot be left blank or set to an empty string.

Parameter	Mandatory	Type	Description
tags_any	No	List<tag>	The return result contains resources corresponding to any tag in this parameter. This parameter contains a maximum of 10 keys, and each key contains a maximum of 10 values. The structure body cannot be missing, and the key cannot be left blank or set to an empty string. Keys must be unique and values of a key must be unique.
not_tags	No	List<tag>	The return result does not contain resources corresponding to all tags in this parameter. This parameter contains a maximum of 10 keys, and each key contains a maximum of 10 values. The structure body cannot be missing, and the key cannot be left blank or set to an empty string. Keys must be unique and values of a key must be unique.

Parameter	Mandatory	Type	Description
not_tags_any	No	List<tag>	The return result does not contain resources corresponding to any tag in this parameter. This parameter contains a maximum of 10 keys, and each key contains a maximum of 10 values. The structure body cannot be missing, and the key cannot be left blank or set to an empty string. Keys must be unique and values of a key must be unique.
limit	No	String	Number of records. This parameter is not available when **action** is set to **count**. The default value is **1000** when **action** is set to **filter**. The maximum value is **1000**, and the minimum value is **1**. The value cannot be a negative number.

Parameter	Mandatory	Type	Description
offset	No	String	Index position. The query starts from the next piece of data specified by the **offset** parameter. This parameter is not required when you query data on the first page. The value in the response body returned for querying data on the previous page will be included in this parameter for querying data on subsequent pages. This parameter is not available when **action** is set to **count**. If **action** is set to **filter**, the value must be a number, and the default value is **0**. The value cannot be a negative number.
action	Yes	String	Operation to be performed. The value can be **filter** or **count**. The value **filter** indicates pagination query. The value **count** indicates that the total number of query results meeting the search criteria will be returned.

Parameter	Mandatory	Type	Description
matches	No	List<match>	Search field. **key** indicates the field to be matched, for example, **resource_name**. **value** indicates the matched value. This field is a fixed dictionary value. Determine whether fuzzy match is required based on different fields. For example, if **key** is **resource_name**, fuzzy search is used by default. If **value** is an empty string, exact match is used. If **key** is **resource_id**, exact match is used. Only **resource_name** is queried at the first phase, and other keys will be queried later.

Table 5-39 tag field description

Parameter	Mandatory	Type	Description
key	Yes	String	Key. It contains a maximum of 127 Unicode characters. It cannot be left empty. (This parameter is not verified in the search process.)

Parameter	Mandatory	Type	Description
values	Yes	List<String>	List of values. A value contains a maximum of 255 Unicode characters. If the values are null, it indicates **any_value**. The relationship between values is OR. By default, only the first value is used for search.

Table 5-40 match field description

Parameter	Mandatory	Type	Description
key	Yes	String	Key. Currently, only **resource_name** is available, indicating a cluster name. Other keys will be supported later.
value	Yes	String	Value. A value contains a maximum of 64 Unicode characters.

Response

Response parameter

The following table describes the response parameters.

Table 5-41 Response parameters

Parameter	Mandatory	Type	Description
resources	Yes	String	Resource list.
total_count	Yes	String	Total number of records.

Table 5-42 Description of the **resource** field data structure

Parameter	Mandatory	Type	Description
resource_id	Yes	String	Resource ID.
resource_detail	Yes	Object	Resource details. The value is a resource object used for extension. The value is left blank by default.
tags	Yes	List<resource_tag>	Tag list. This parameter is left blank by default if there is no tag.
resource_name	Yes	string	Resource name. This parameter is left blank by default if there is no resource name.

Table 5-43 Description of the **resource_tag** field data structure

Parameter	Mandatory	Type	Description
key	Yes	String	Key. A tag key consists of a maximum of 36 characters, including A-Z, a-z, 0-9, hyphens (-), and underscores (_).
value	Yes	String	Value. The value consists of a maximum of 43 characters, including A-Z, a-z, 0-9, periods (.), hyphens (-), and underscores (_), and can be an empty string.

Example

- Request example

Filter request example

```
POST /{version}/{project_id}/{resource_type}/resource_instances/action
```

Request body when **action** is set to **filter**

```
{
  "offset": "100",
  "limit": "100",
"action": "filter",
  "matches":[
```

```
{
        "key": "resource_name",
        "value": "streamA"
       }
],
   "not_tags": [
    {
      "key": "key1",
      "values": [
        "value1",
        "value2"
      ]
    }
  ],
  "tags": [
    {
      "key": "key1",
      "values": [
        "value1",
        "value2"
      ]
    }
  ],
  "tags_any": [
    {
      "key": "key1",
      "values": [
        "value1",
        "value2"
      ]
    }
  ],
"not_tags_any": [
    {
      "key": "key1",
      "values": [
        "value1",
        "value2"
      ]
    }
  ]
}
```

Request body when **action** is set to **count**

```
{
  "action": "count",
  "not_tags": [
    {
      "key": "key1",
      "values": [
        "value1",
        "value2"
      ]
    }
  ],
  "tags": [
    {
      "key": "key1",
      "values": [
        "value1",
        "value2"
      ]
    },
  {
      "key": "key2",
      "values": [
        "value1",
        "value2"
      ]
```

```
    }
  ],
  "tags_any": [
    {
      "key": "key1",
      "values": [
        "value1",
        "value2"
      ]
    }
  ],
"not_tags_any": [
    {
      "key": "key1",
      "values": [
        "value1",
        "value2"
      ]
    }
   ],
"matches":[
{
        "key": "resource_name",
        "value": "streamA"
       }
]
}
```

- Response example

Response body when **action** is set to **filter**

```
    {
      "resources": [
         {
            "resource_detail": null,
            "resource_id": "cdfs_cefs_wesas_12_dsad",
            "resource_name": "streamA",
            "tags": [
                {
                   "key": "key1",
                   "value": "value1"
                },
                {
                   "key": "key2",
                   "value": "value1"
                }
             ]
         }
       ],
      "total_count": 1000
}
```

Response body when **action** is set to **count**

```
{
       "total_count": 1000
}
```

Status Code

Table 5-44 lists the status code of this API.

Table 5-44 Status code

Status Code	Description
200	OK

6 Common Parameters

6.1 Common Request Message Headers

Table 6-1 describes the DIS request headers.

Table 6-1 Common request message headers

Header	Mandatory	Type	Description
Content-Type	Yes	String	Type of the resource content. For example: application/json Default value: none
X-Sdk-Date	Yes	String	Time to send a request. The time format is *YYYYMMDD*'**T**'*HHMMSS*'**Z**'. For example: 20180820T101459Z Default value: none
Authorization	Yes	String	Authorization information in a request. SDK-HMAC-SHA256 Credential=ZIRRKMTWPTQFQI1WKNKB/20150907/eu-de/ec2/sdk_request, SignedHeaders=content-type;host;x-sdk-date, Signature=55741b610f3c9fa3ae40b5a8021ebf7ebc2a28a603fc62d25cb3bfe6608e1994 Default value: none
Host	Yes	String	Requested service information. Default value: none

6.2 Common Response Message Headers

Table 6-2 describes the DIS response headers.

Table 6-2 Common response message headers

Header	Type	Description
Content-Type	String	Type of the resource content. For example: application/json Default value: none
Date	String	DIS system response time. For example: Mon, 15 Jul 2013 21:08:05 GMT Default value: none

6.3 Error Codes

If API calling fails, a response with HTTP status code 4*xx* or 5*xx* is returned. The returned message body contains the specific error code and error information. The response also includes specific error code and error information in the message body, as described in the following table.

Example:

{

"errorCode": "DIS.4301",

"message": "Stream does not exist. [test][6332998f84ac4c13a83db055da33cb66]"

}

Error Code	Description
DIS.4100	Authorization error.
DIS.4101	Empty Authorization header.
DIS.4102	Incorrectly parsed authorization header.
DIS.4103	Empty X-Sdk-Date header.
DIS.4104	Incorrectly parsedX-Sdk-Date header.
DIS.4105	Invalid X-Sdk-Date header.
DIS.4106	Empty AcessKey header.
DIS.4107	Invalid AcessKey header.
DIS.4108	Empty ServiceName header.
DIS.4109	The Authorization header must contain the following field: {Credential,SignedHeaders,Signature;}
DIS.4110	Empty Signature header.
DIS.4111	Invalid Region header.

Error Code	Description
DIS.4112	Invalid authorization request.
DIS.4113	Empty Token header.
DIS.4114	Invalid Token header.
DIS.4115	Unknown X-**-Target.
DIS.4116	Invalid RBAC.
DIS.4117	Invalid Project Id.
DIS.4200	Invalid request.
DIS.4201	Invalid partition_id.
DIS.4202	Empty request.
DIS.4203	Invalid monitoring period.
DIS.4204	The monitoring period cannot be longer than 3 days.
DIS.4208	Invalid MRS cluster.
DIS.4209	Invalid metrics label.
DIS.4215	Invalid cursor type.
DIS.4216	Invalid sequence_number.
DIS.4217	Invalid partition cursor.
DIS.4218	The record amount exceeds 1.
DIS.4219	The file is constantly resent.
DIS.4220	The block whose sequence number is %s needs to be resent.
DIS.4221	Block seq %s is expected.
DIS.4222	Block seq %s is expected.
DIS.4223	The file size exceeds the limit.
DIS.4224	The sequence number is out of range.
DIS.4225	Expired partition cursor.
DIS.4226	Partition iterator error. Try to obtain a partition iterator again.
DIS.4300	Request error.
DIS.4301	The stream does not exist.
DIS.4302	The partition does not exist.

Error Code	Description
DIS.4303	Exceeded traffic control limit.
DIS.4304	Expired request time.
DIS.4305	Too many stream requests.
DIS.4306	The bucket does not exist.
DIS.4307	The stream already exists.
DIS.4308	Insufficient quota.
DIS.4309	The IP address is on the black list.
DIS.4310	OBS access error.
DIS.4329	Exceeded application quota.
DIS.4330	The application already exists.
DIS.4331	The application is being used.
DIS.4332	The application is not found.
DIS.4335	Invalid IAM agency.
DIS.4336	Invalid HDFS path.
DIS.4337	The DLI database does not exist.
DIS.4338	The DLI table does not exist.
DIS.4341	The CloudTable cluster does not exist.
DIS.4342	The CloudTable table does not exist
DIS.4343	The CloudTable table family does not exist.
DIS.4345	Invalid CloudTable schema.
DIS.4348	Invalid CloudTable openTSDB schema.
DIS.4350	Invalid DWS cluster.
DIS.4351	Invalid KMS userKey.
DIS.4354	The transfer task does not exist.
DIS.4355	The transfer task already exists.
DIS.4357	Exceeded transfer task quota.
DIS.4358	The stream supports specific transfer tasks. Check the data type of the stream.
DIS.4360	Invalid data schema.
DIS.4601	The number of resource tags has reached the maximum.

Error Code	Description
DIS.4602	Invalid resource type.
DIS.4603	The resource does not exist.
DIS.4604	The key does not exist.
DIS.4605	The action is not supported.
DIS.5000 to DIS.5999	System error. **NOTE** Contact technical support to handle system errors.

6.4 Status Codes

A status code is an HTTPS response issued by DIS to indicate whether an API request has been successfully completed.

Status Code	Status	Description
100	Continue	The server has received the initial part of the request and the client should continue to send the remaining part. It is issued on a provisional basis while request processing continues. It alerts the client to wait for a final response.
101	Switching Protocols	The requester has asked the server to switch protocols and the server has agreed to do so. The target protocol must be more advanced than the source protocol. For example, the current HTTP protocol is switched to a later version of HTTP.
200	OK	The server has successfully processed the request.
201	Created	The request has been fulfilled, resulting in the creation of a new resource.
202	Accepted	The request has been accepted for processing, but the processing has not been completed.
203	Non-Authoritative Information	The server successfully processed the request, but is returning information that may be from another source.

Status Code	Status	Description
204	NoContent	The server has successfully processed the request, but does not return any content. The status code is returned in response to an HTTPS OPTIONS request.
205	Reset Content	The server has successfully processed the request, but does not return any content. Unlike a 204 response, this response requires that the requester reset the content.
206	Partial Content	The server has successfully processed a part of the GET request.
300	Multiple Choices	There are multiple options for the requested resource. For example, this code could be used to present a list of resource characteristics and addresses from which the client such as a browser may choose.
301	Moved Permanently	This and all future requests should be permanently directed to the given URI indicated in this response.
302	Found	The requested resource was temporarily moved.
303	See Other	The response to the request can be found under another URI using a GET or POST method.
304	Not Modified	The requested resource has not been modified. In such a case, there is no need to retransmit the resource since the client still has a previously-downloaded copy.
305	Use Proxy	The requested resource is available only through a proxy.
306	Unused	This HTTP status code is no longer used.
400	BadRequest	The request is invalid. The client should modify the request instead of re-initiating it.
401	Unauthorized	The authorization information provided by the client is incorrect or invalid.
402	Payment Required	Reserved for future use.
403	Forbidden	The server has received the request and understood it, but the server is refusing to respond to it. The client should modify the request instead of re-initiating it.

Status Code	Status	Description
404	NotFound	The requested resource could not be found. The client should modify the request instead of re-initiating it.
405	MethodNotAllowed	A request method is not supported for the requested resource. The client should modify the request instead of re-initiating it.
406	Not Acceptable	The server could not fulfill the request according to the content characteristics of the request.
407	Proxy Authentication Required	This code is similar to 401, but indicates that the client must first authenticate itself with the proxy.
408	Request Time-out	The server timed out waiting for the request. The client may re-initiate the request without modifications at any later time.
409	Conflict	The request could not be processed due to a conflict in the request. For example, an edit conflict between multiple simultaneous updates or the resource that the client attempts to create already exists.
410	Gone	The requested resource has been deleted permanently and will not be available again. The status code indicates that the requested resource has been deleted permanently.
411	Length Required	The server refused to process the request because the request does not specify the length of its content.
412	Precondition Failed	The server does not meet one of the preconditions that the requester puts on the request.
413	Request Entity Too Large	The request is larger than the server is willing or able to process. The server may close the connection to prevent the client from continuing the request. If the server temporarily cannot process the request, the response will contain a Retry-After header field.
414	Request-URI Too Large	The URI provided was too long for the server to process.
415	Unsupported Media Type	The server does not support the media type in the request.
416	Requested range not satisfiable	The requested range is invalid.

Status Code	Status	Description
417	Expectation Failed	The server fails to meet the requirements of the Expect request-header field.
422	UnprocessableEntity	The request was well-formed but was unable to be followed due to semantic errors.
429	TooManyRequests	The client has sent more requests than its rate limit is allowed within a given amount of time, or the server has received more requests than it is able to process within a given amount of time. In this case, it is advisable for the client to re-initiate requests after the time specified in the Retry-After header of the response expires.
441	Authentication Error	Authentication fails.
500	InternalServerError	The server is able to receive the request but it could not understand the request.
501	Not Implemented	The server does not support the requested function.
502	Bad Gateway	The server was acting as a gateway or proxy and received an invalid request from a remote server.
503	ServiceUnavailable	The requested service is invalid. The client should modify the request instead of re-initiating it.
504	ServerTimeout	The server could not return a timely response. The response will reach the client only if the request carries a timeout parameter.
505	HTTP Version not supported	The server does not support the HTTP protocol version used in the request.

A Change History

Release Date	What's New
2018-10-25	Accepted in OTC 3.2.
2018-09-29	Modified the following contents: Tag management APIs were added in **Tag Management APIs**.
2018-09-26	Modified the following contents: A status code was added in **Status Codes**.
2018-03-16	This issue is the first official release.