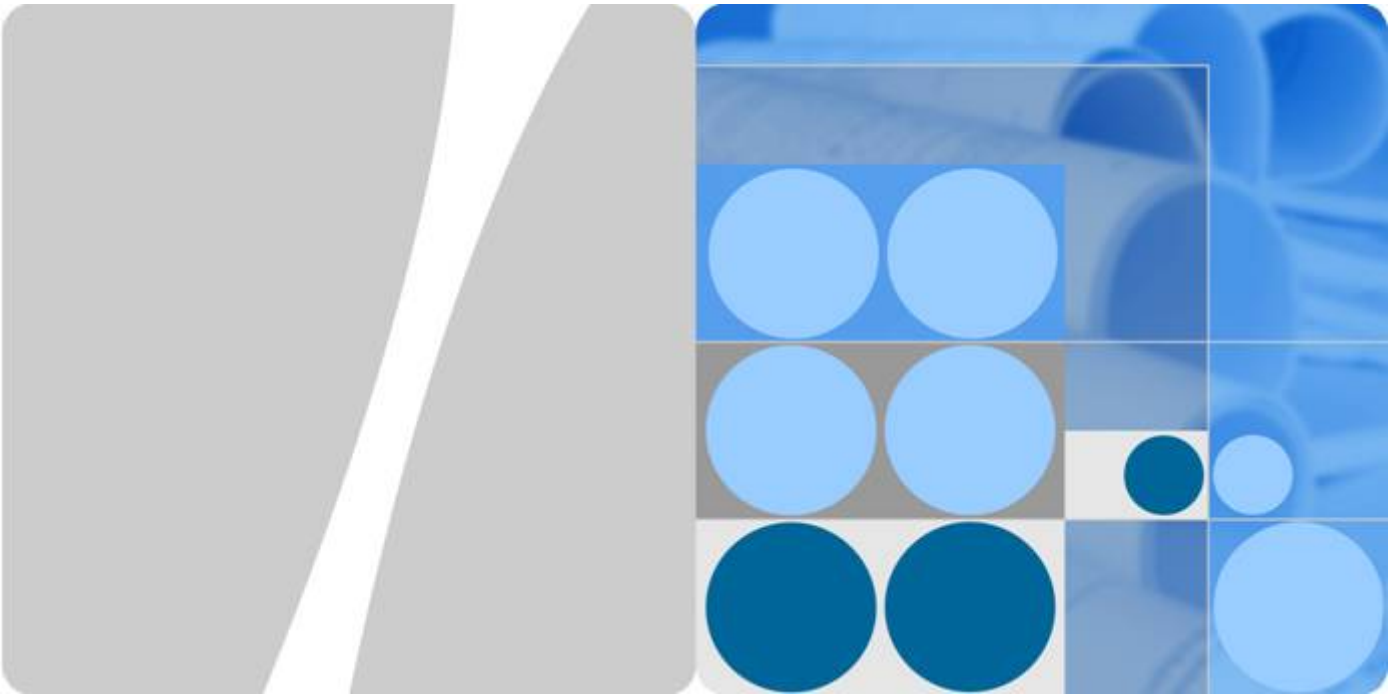

Cloud Stream Service

Stream Ecosystem Development Guide

Issue 12

Date 2019-02-26

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

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 Introduction to the Stream Ecosystem

Built on Flink and Spark, the stream ecosystem is fully compatible with the open-source Flink, Storm, and Spark APIs. It is enhanced in features and improved in performance to provide easy-to-use Cloud Stream Service (CS) with low latency and high throughput.

The CS ecosystem consists of cloud service ecosystems and open-source ecosystems.

- Cloud service ecosystems

  By using Stream SQL, CS can be interconnected with other services, such as Data Ingestion Service (DIS), Object Storage Service (OBS), CloudTable Service (CloudTable), MapReduce Service (MRS), Relational Database Service (RDS), Simple Message Notification (SMN), and Distributed Cache Service (DCS). You can directly use SQL to read data from or write data into these services.

- Open-source ecosystem

  After connections to other VPCs are established through VPC peering connections, you can access all data sources and output targets (such as Kafka, HBase, and Elasticsearch) supported by Flink and Spark in your exclusive CS cluster.

2 Data Formats

CS can work with the following data formats: CSV, JSON, EMAIL, ORC, BLOB, XML, and CarbonData. The following table lists the supported source and sink streams of corresponding data formats. You can click the links in the following table to view examples corresponding to different data formats.

Table 2-1 Supported source and sink streams of corresponding data formats

Data Format	Supported Source Stream	Supported Sink Stream
CSV	● DIS Source Stream ● OBS Source Stream ● Open-Source Kafka Source Stream	● DIS Sink Stream ● OBS Sink Stream ● DWS Sink Stream (OBS-based Dumping) ● API Gateway (APIG) Sink Stream ● Open-Source Kafka Sink Stream
JSON	● DIS Source Stream ● OBS Source Stream ● MRS Kafka Source Stream ● Open-Source Kafka Source Stream	● DIS Sink Stream ● OBS Sink Stream ● MRS Kafka Sink Stream ● APIG Sink Stream ● Open-Source Kafka Sink Stream
EMAIL	● DIS Source Stream	-
ORC	-	● OBS Sink Stream ● DWS Sink Stream (OBS-based Dumping)
BLOB	● DIS Source Stream ● MRS Kafka Source Stream ● Open-Source Kafka Source Stream	-
XML	DIS Source Stream	-

Data Format	Supported Source Stream	Supported Sink Stream
CarbonData	-	**OBS Sink Stream**

3 Cloud Ecosystem: DIS

3.1 DIS Source Stream

Overview

Create a source stream to read data from DIS. DIS accesses user data and CS reads data from the DIS stream as input data for jobs. This cloud ecosystem is applicable to scenarios where data out of cloud services is imported into cloud services for filtering, real-time analysis, monitoring and reporting, and dumping.

DIS addresses the challenge of transmitting data outside cloud services to cloud services. DIS builds data intake streams for custom applications capable of processing or analyzing streaming data. DIS continuously captures, transmits, and stores terabytes of data from hundreds of thousands of sources every hour, such as logs, Internet of Things (IoT) data, social media feeds, website clickstreams, and location-tracking events. For more information about DIS, see the **Data Ingestion Service User Guide**.

Syntax

Syntax

CREATE SOURCE STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "dis",region = "",channel = "",partition_count = "",encode = "",field_delimiter = "",offset= "")(TIMESTAMP BY timeindicator (',' timeindicator)?);timeindicator:PROCTIME '.' PROCTIME| ID '.' ROWTIME

Description

Table 3-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Data source type. Value **dis** indicates that the data source is DIS.
region	Yes	Region where DIS for storing the data is located.

Parameter	Mandatory	Description
channel	Yes	Name of the DIS stream where data is located.
partition_count	No	Number of partitions of the DIS stream where data is located. This parameter and **partition_range** cannot be configured at the same time. If this parameter is not specified, data of all partitions is read by default.
partition_range	No	Range of partitions of a DIS stream, data in which is ingested by the CS job. This parameter and **partition_count** cannot be configured at the same time. If this parameter is not specified, data of all partitions is read by default. If this parameter is set to [2:5], the CS job will ingest data in partitions 2 and 5.
encode	Yes	Data encoding format. Available options include **csv**, **json**, **xml**, **email**, and **blob**. **NOTE** ● **field_delimiter** must be specified if this parameter is set to **csv**. ● **json_config** must be specified if this parameter is set to **json**. ● **xml_config** must be specified if this parameter is set to **xml**. ● **email_key** must be specified if this parameter is set to **email**. ● If this parameter is set to **blob**, the received data is not parsed, only one stream attribute exists, and the data format is ARRAY[TINYINT].
field_delimiter	No	Attribute delimiter. You can set this parameter only when **encode** is set to **csv**. The default value is a comma (**,**).
quote	No	Quoted symbol in a data format. The attribute delimiters between two quoted symbols are treated as common characters. ● If double quotation marks are used as the quoted symbol, set this parameter to "\u005c\u0022" for character conversion. ● If a single quotation mark is used as the quoted symbol, set this parameter to a comma (,). **NOTE** After this parameter is specified, ensure that each field does not contain quoted symbols or contains an even number of quoted symbols. Otherwise, parsing will fail.

Parameter	Mandatory	Description
json_config	No	If **encode** is set to **json**, you need to set this parameter to specify the mapping between the JSON field and the stream definition field. An example of the format is as follows: field1=data_json.field1; field2=data_json.field2.
xml_config	No	If **encode** is set to **xml**, you need to set this parameter to specify the mapping between the xml field and the stream definition field. An example of the format is as follows: field1=data_xml.field1; field2=data_xml.field2.
email_key	No	If **encode** is set to **email**, you need to set the parameter to specify the information to be extracted. You need to list the key values that correspond to stream definition fields. Multiple key values are separated by commas (,), for example, "Message-ID, Date, Subject, body". There is no keyword in the email body and CS specifies "body" as the keyword.
offset	No	● If data is imported to the DIS stream after the job is started, this parameter will become invalid. ● If the job is started after data is imported to the DIS stream, you can set the parameter as required. For example, if **offset** is set to **100**, CS starts from the 100th data record in DIS.
start_time	No	Start time for reading DIS data. ● If this parameter is specified, CS reads data read from the specified time. The parameter value is in the format of yyyy-MM-dd HH:mm:ss. ● If neither **start_time** nor **offset** is specified, CS reads the latest data. ● If **start_time** is not specified but **offset** is specified, CS reads data from the data record specified by **offset**.

Parameter	Mandatory	Description
timeindicator	No	Timestamp added in the source stream. The value can be **processing time** or **event time**. **NOTE** ● If this parameter is set to **processing time**, the format is **proctime.proctime**. In this case, an attribute proctime will be added to the original attribute field. If there are three attributes in the original attribute field, four attributes will be exported after this parameter is set to **processing time**. However, the attribute length remains unchanged if the rowtime attribute is specified. ● If this parameter is set to **event time**, you can select an attribute in the stream as the timestamp. The format is **attr_name.rowtime**. ● This parameter can be simultaneously set to **processing time** and **event time**.
enable_checkpoint	No	Whether to enable the checkpoint function. Value **true** indicates to enable the checkpoint function, and value **false** indicates to disable the checkpoint function. The default value is **false**.
checkpoint_app_name	No	ID of a DIS consumer. If a DIS stream is consumed by different jobs, you need to configure the consumer ID for each job to avoid checkpoint confusion.
checkpoint_interval	No	Interval of checkpoint operations on the DIS source operator. The value is in the unit of seconds. The default value is **60**.

Precautions

The attribute type used as the timestamp must be long or timestamp.

Example

- In CSV encoding format, CS reads data from the DIS stream and records it as codes in CSV format. The codes are separated by commas (,).

```
CREATE SOURCE STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT,
  car_timestamp LONG
)
WITH (
  type = "dis",
  region = "cn-north-1" ,
  channel = "csinput",

  encode = "csv",
```



```
  field_delimiter = ","
)TIMESTAMP BY car_timestamp.rowtime;
```

- In JSON encoding format, CS reads data from the DIS stream and records it as codes in JSON format. For example, {"car":{"car_id":"ZJA710XC", "car_owner":"coco", "car_age":5, "average_speed":80, "total_miles":15000, "car_timestamp":1526438880}}

```
CREATE SOURCE STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT,
  car_timestamp LONG
)
WITH (
  type = "dis",
  region = "cn-north-1" ,
  channel = "csinput",

  encode = "json",
  json_config = "car_id=car.car_id;car_owner
=car.car_owner;car_age=car.car_age;average_speed
=car.average_speed ;total_miles=car.total_miles;"
)TIMESTAMP BY car_timestamp.rowtime;
```

- In XML encoding format, CS reads data from the DIS stream and records it as codes in XML format.

```
CREATE SOURCE STREAM person_infos (
    pid BIGINT,
    pname STRING,
    page int,
    plocation STRING,
    pbir DATE,
    phealthy BOOLEAN,
    pgrade ARRAY[STRING]
)
WITH (
  type = "dis",
  region = "cn-north-1" ,
  channel = "dis-cs-input",
  encode = "xml",
  field_delimiter = ",",
  xml_config =
"pid=person.pid;page=person.page;pname=person.pname;plocation=person.plocation
;pbir=person.pbir;pgrade=person.pgrade;phealthy=person.phealthy"
);
```

An example of XML data is as follows:

```
<?xml version="1.0" encodeing="utf-8"?>

<root>
  <person>
    <pid>3623051990100250 42</pid>
    <pname>xiaoming</pname>
    <page>28</page>
    <plocation>Shangdu County, Jining District, Ulanchap, Inner Mongolia</
plocation>
    <pbir>1990-10-02</pbir>
    <phealthy>true</phealthy>
    <pgrade>[A,B,C]</pgrade>
  </person>
</root>
```

- In EMAIL encoding format, CS reads data from the DIS stream and records it as a complete Email.

```
CREATE SOURCE STREAM email_infos (
Event_ID String,
Event_Time Date,
Subject String,
From_Email String,
```



```
To_EMAIL String,
CC_EMAIL Array[String],
BCC_EMAIL String,
MessageBody String,
Mime_Version String,
Content_Type String,
charset String,
Content_Transfer_Encoding String
)
WITH (
type = "dis",
region = "cn-north-1" ,
channel = "csinput",

encode = "email",
email_key = "Message-ID, Date, Subject, From, To, CC, BCC, Body, Mime-
Version, Content-Type, charset, Content_Transfer_Encoding"
);
```

An example of email data is as follows:

```
Message-ID: <20090629183903250425 4@sample.com>
Date: Fri, 11 May 2001 09:54:00 -0700 (PDT)
From: zhangsan@sample.com
To: lisi@sample.com, wangwu@sample.com
Subject:  "Hello World"
Cc: lilei@sample.com, hanmei@sample.com
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Bcc: jack@sample.com, lily@sample.com
X-From: Zhang San
X-To: Li Si, Wang Wu
X-cc: Li Lei, Han Mei
X-bcc:
X-Folder: \Li_Si_June2001\Notes Folders\Notes inbox
X-Origin: Lucy
X-FileName: sample.nsf

Dear Associate / Analyst Committee:

Hello World!

Thank you,

Associate / Analyst Program
zhangsan
```

3.2 DIS Sink Stream

Overview

CS writes the job output data into DIS. This cloud ecosystem is applicable to scenarios where data is filtered and imported to the DIS stream for future processing.

DIS addresses the challenge of transmitting data outside cloud services to cloud services. DIS builds data intake streams for custom applications capable of processing or analyzing streaming data. DIS continuously captures, transmits, and stores terabytes of data from hundreds of thousands of sources every hour, such as logs, Internet of Things (IoT) data, social media feeds, website clickstreams, and location-tracking events. For more information about DIS, see the **Data Ingestion Service User Guide**.

Syntax

Syntax

CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "dis",region = "",channel = "",partition_key = "",encode= "",field_delimiter= "");

Description

Table 3-2 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **dis** indicates that data is stored to DIS.
region	Yes	Region where DIS for storing the data is located.
channel	Yes	DIS stream.
partition_key	No	Group primary key. Multiple primary keys are separated by commas (,). If this parameter is not specified, data is randomly written to DIS partitions.
encode	Yes	Data encoding format. The value can be **csv** and **json**. **NOTE** ● If the encoding format is **csv**, you need to configure attribute separators. ● If the encoding format is **json**, you need to configure **enableOutputNull** to control whether to generate an empty field. For details, see the examples.
field_delimiter	Yes	Separator used to separate every two attributes. ● This parameter needs to be configured if the CSV encoding format is adopted. It can be user-defined, for example, a comma (**,**). ● This parameter is not required if the JSON encoding format is adopted.
json_config	No	If **encode** is set to **json**, you can set this parameter to specify the mapping between the JSON field and the stream definition field. An example of the format is as follows: field1=data_json.field1; field2=data_json.field2.

Parameter	Mandatory	Description
enableOutputNull	No	If **encode** is set to **json**, you need to specify this parameter to control whether to generate an empty field. If this parameter is set to **true**, an empty field (the value is **null**) is generated. If set to **false**, no empty field is generated.

Precautions

None

Example

- CSV: Data is written to the DIS stream and encoded using CSV. CSV fields are separated by commas (,). If there are multiple partitions, car_owner is used as the key to distribute data to different partitions. An example is as follows: "ZJA710XC", "lilei", "BMW", 700000.

```
CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
WITH (
  type = "dis",
  region = "cn-north-1" ,
  channel = "csoutput",
  encode = "csv",
  field_delimiter = ","
);
```

- JSON: Data is written to the DIS stream and encoded using JSON. If there are multiple partitions, car_owner and car_brand are used as the keys to distribute data to different partitions. If **enableOutputNull** is set to **true**, an empty field (the value is **null**) is generated. If set to **false**, no empty field is generated. An example is as follows: "car_id ":"ZJA710XC", "car_owner ":"lilei", "car_brand ":"BMW", "car_price ":700000.

```
CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
WITH (
  type = "dis",
  channel = "csoutput",
  region = "cn-north-1" ,
  partition_key = "car_owner,car_brand",
  encode = "json",
  enable_output_null = "false"
);
```


4 Cloud Ecosystem: OBS

4.1 OBS Source Stream

Overview

Create a source stream to obtain data from OBS. CS reads data stored by users in OBS as input data for jobs. OBS applies to various scenarios, such as big data analysis, cloud-native application program data, static website hosting, backup/active archive, and deep/cold archive.

OBS is an object-based storage service. It provides massive, secure, highly reliable, and low-cost data storage capabilities. For more information, see the **Object Storage Service Console Operation Guide**.

Syntax

Syntax

CREATE SOURCE STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "obs",region = "",bucket = "",object_name = "",row_delimiter = "\n",field_delimiter = '',version_id = "")(TIMESTAMP BY timeindicator (',' timeindicator)?);timeindicator:PROCTIME '.' PROCTIME| ID '.' ROWTIME

Description

Table 4-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Data source type. Value **obs** indicates that the data source is OBS.
region	Yes	Region to which OBS belongs.
bucket	Yes	Name of the OBS bucket where data is located.

Parameter	Mandatory	Description
object_name	Yes	Name of the object stored in the OBS bucket where data is located.
row_delimiter	Yes	Separator used to separate every two rows.
field_delimiter	Yes	Separator used to separate every two attributes. ● You can set this parameter only when **encode** is set to **csv**. The default value is a comma (**,**). ● This parameter is not required if the JSON encoding format is adopted.
quote	No	Quoted symbol in a data format. The attribute delimiters between two quoted symbols are treated as common characters. ● If double quotation marks are used as the quoted symbol, set this parameter to "\u005c\u0022" for character conversion. ● If a single quotation mark is used as the quoted symbol, set this parameter to a comma (,). **NOTE** After this parameter is specified, ensure that each field does not contain quoted symbols or contains an even number of quoted symbols. Otherwise, parsing will fail.
version_id	No	Version number. This parameter is optional and required only when the OBS bucket or object has version settings.

Parameter	Mandatory	Description
timeindicator	No	Timestamp added in the source stream. The value can be **processing time** or **event time**. **NOTE** ● If this parameter is set to **processing time**, the format is **proctime.proctime**. In this case, an attribute proctime will be added to the original attribute field. If there are three attributes in the original attribute field, four attributes will be exported after this parameter is set to **processing time**. However, the attribute length remains unchanged if the rowtime attribute is specified. ● If this parameter is set to **event time**, you can select an attribute in the stream as the timestamp. The format is **attr_name.rowtime**. ● This parameter can be simultaneously set to **processing time** and **event time**.

Precautions

The attribute type used as the timestamp must be long or timestamp.

Example

The **input.csv** file is read from the OBS bucket. Rows are separated by **'\n'** and columns are separated by **','**.

```
CREATE SOURCE STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
WITH (
  type = "obs",
  bucket = "obssource",
  region = "cn-north-1" ,
  object_name = "input.csv",
  row_delimiter = "\n",
  field_delimiter = ",",

)TIMESTAMP BY car_timestamp.rowtime;
```

4.2 OBS Sink Stream

Overview

Create a sink stream to export CS data to OBS. CS can export the job analysis results to OBS. OBS applies to various scenarios, such as big data analysis, cloud-native application program data, static website hosting, backup/active archive, and deep/cold archive.

OBS is an object-based storage service. It provides massive, secure, highly reliable, and low-cost data storage capabilities. For more information, see the **Object Storage Service Console Operation Guide**.

Precautions

Before data exporting, check the version of the OBS bucket. The OBS sink stream supports data exporting to an OBS bucket running OBS 3.0 or a later version.

Syntax

Syntax

```
CREATE SINK STREAM stream_id (
   attr_name attr_type (',' attr_name attr_type)*
)WITH (
            type = "obs",
            region = "",
            encode = "",
            field_delimiter = "",
            row_delimiter = "",
            obs_dir = "",
            file_prefix = "",
            rolling_size = "",
            rolling_interval = "",
            quote = "",
            array_bracket = "",
            append = "",
            max_record_num_per_file = "",
            dump_interval = "",
            dis_notice_channel = "",
            max_record_num_cache = "",
            carbon_properties = ""
)
```

Description

Table 4-2 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **obs** indicates that data is stored to OBS.
region	Yes	Region to which OBS belongs.
encode	Yes	Encoding format. Currently, formats CSV, JSON, ORC, and CarbonData are supported.
field_delimiter	No	Separator used to separate every two attributes. ● This parameter is mandatory only when the CSV encoding format is adopted. If this parameter is not specified, the default separator comma (,) is used. ● This parameter does not need to be configured if the CarbonData, JSON, or ORC encoding format is adopted.

Parameter	Mandatory	Description
row_delimiter	No	Row delimiter. This parameter does not need to be configured if the CarbonData or ORC encoding format is adopted.
json_config	No	If **encode** is set to **json**, you can set this parameter to specify the mapping between the JSON field and the stream definition field. An example of the format is as follows: field1=data_json.field1;field2=data_json.field2.
obs_dir	Yes	Directory for storing files. The directory is in the format of {Bucket name}/{Directory name}, for example, obs-a1/dir1/subdir.
file_prefix	No	Prefix of the data export file name. The generated file is named in the format of file_prefix.*x*, for example, file_prefix.1 and file_prefix.2. If this parameter is not specified, the file prefix is **temp** by default. This parameter is not applicable to CarbonData files.
rolling_size	No	Maximum size of a file. **NOTE** ● One or both of **rolling_size** and **rolling_interval** must be configured. ● When the size of a file exceeds the specified size, a new file is generated. ● The unit can be KB, MB, or GB. If no unit is specified, the byte unit is used. ● This parameter does not need to be configured if the ORC or CarbonData encoding format is adopted.
rolling_interval	No	Time mode, in which data is saved to the corresponding directory. **NOTE** ● One or both of **rolling_size** and **rolling_interval** must be configured. ● After this parameter is specified, data is written to the corresponding directories according to the output time. ● The parameter value can be in the format of yyyy/MM/dd/HH/mm, which is case sensitive. The minimum unit is minute. If this parameter is set to **yyyy/MM/dd/HH**, data is written to the directory that is generated at the hour time. For example, data generated at 2018-09-10 16:40 will be written to the **{obs_dir}/2018-09-10_16** directory. ● If both **rolling_size** and **rolling_interval** are set, a new file is generated when the size of a single file exceeds the specified size in the directory corresponding to each time point.

Parameter	Mandatory	Description
quote	No	Modifier, which is added before and after each attribute only when the CSV encoding format is adopted. You are advised to use invisible characters, such as **u0007**, as the parameter value.
array_bracket	No	Array bracket, which can be configured only when the CSV encoding format is adopted. The available options are **()**, **{}**, and **[]**. For example, if you set this parameter to **{}**, the array output format is {a1, a2}.
append	No	The value can be **true** or **false**. The default value is **true**. If OBS does not support the append mode, set this parameter to **false**. If **Append** is set to **false**, **max_record_num_per_file** and **dump_interval** must be set.
max_record_num_per_file	No	Maximum number of records in a file. This parameter needs to be configured only when the ORC or CarbonData encoding format is adopted. After this parameter is specified, a new file is generated to store extra data records after the number of records stored in a file reaches the allowed quantity.
dump_interval	No	Triggering period. This parameter needs to be configured when the ORC encoding format is adopted or notification to DIS is enabled. ● If the ORC encoding format is specified, this parameter indicates that files will be uploaded to OBS when the triggering period arrives even if the number of file records does not reach the maximum value. ● In notification to DIS is enabled, this parameter specifies that a notification is sent to DIS every period to indicate that no more files will be generated in the directory.
dis_notice_channel	No	DIS stream where CS sends the record that contains the OBS directory CS periodically sends the DIS stream a record, which contains the OBS directory, indicating that no more new files will be generated in the directory.
max_record_num_cache	No	Maximum number of cache records. This parameter can be set only when the CarbonData encoding format is adopted. The minimum value of this parameter cannot be less than that of **max_record_num_per_file**. The default value is **max_record_num_per_file**.

Parameter	Mandatory	Description
carbon_properties	No	Carbon attribute. This field can be configured only when the CarbonData encoding format is adopted. The value is in the format of k1=v1, k2=v2. All configuration items supported by the withTableProperties function in carbon-sdk are supported. In addition, the configuration items **IN_MEMORY_FOR_SORT_DATA_IN_MB** and **UNSAFE_WORKING_MEMORY_IN_MB** are supported.

Example

- Export the **car_infos** data to the **obs-sink** bucket in OBS. The output directory is **car_infos**. The output file uses **greater_30** as the file name prefix. The maximum size of a single file is 100 MB. If the data size exceeds 100 MB, another new file is generated. The data is encoded in CSV format, the comma (,) is used as the attribute delimiter, and the line break is used as the line separator.

```
CREATE SINK STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
WITH (
  type = "obs",
  encode = "csv",
  region = "cn-north-1" ,
  field_delimiter = ",",
  row_delimiter = "\n",
  obs_dir = "obs-sink/car_infos",
   file_prefix = "greater_30",
   rolling_size = "100m"
);
```

- Example of the CarbonData encoding format

```
CREATE SINK STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
WITH (
  type = "obs",
  region = "cn-north-1" ,
  encode = "carbondata",
  obs_dir = "cs-append-2/carbondata",
  max_record_num_per_file = "1000",
  max_record_num_cache = "2000",
  dump_interval = "60",
  ROLLING_INTERVAL = "yyyy/MM/dd/HH/mm",
  dis_notice_channel = "dis-notice",
  carbon_properties = "long_string_columns=MessageBody,
IN_MEMORY_FOR_SORT_DATA_IN_MB=512"
);
```

- Example of the ORC encoding format

```
CREATE SINK STREAM car_infos (
  car_id STRING,
  car_owner STRING,
```



```
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
WITH (
  type = "obs",
  region = "cn-north-1" ,
  encode = "orc",
  obs_dir = "cs-append-2/obsorc",
  FILE_PREFIX = "es_info",
  max_record_num_per_file = "100000",
  dump_interval = "60"
);
```


5 Cloud Ecosystem: CloudTable

5.1 HBase Source Stream

Overview

Create a source stream to obtain data from HBase of CloudTable as input data of the job. HBase is a column-oriented distributed cloud storage system that features enhanced reliability, excellent performance, and elastic scalability. It applies to the storage of massive amounts of data and distributed computing. You can use HBase to build a storage system capable of storing TB- or even PB-level data. With HBase, you can filter and analyze data with ease and get responses in milliseconds, rapidly mining data value. CS can read data from HBase for filtering, analysis, and data dumping.

CloudTable is a distributed, scalable, and fully-hosted key-value data storage service based on Apache HBase. It provides CS with high-performance random read and write capabilities, which are helpful when applications need to store and query a massive amount of structured data, semi-structured data, and time series data. CloudTable applies to IoT scenarios and storage and query of massive volumes of key-value data. For more information about CloudTable, see the **CloudTable Service User Guide**.

Syntax

Syntax

CREATE SOURCE STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "cloudtable",region = "",cluster_id = "",table_name = "",table_columns = "")(TIMESTAMP BY timeindicator (',' timeindicator)?);timeindicator:PROCTIME '.' PROCTIME| ID '.' ROWTIME

Description

Table 5-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Data source type. Value **CloudTable** indicates that the data source is CloudTable.
region	Yes	Region to which CloudTable belongs.
cluster_id	Yes	ID of the cluster to which the data table to be read belongs. For details about how to view the ID of the CloudTable cluster, see section "**Viewing Basic Cluster Information**" in the **CloudTable Service User Guide**.
table_name	Yes	Name of the table where data is to be read. If namespace needs to be specified, the value can be namespace_name:table_name.
table_columns	Yes	Column to be read. The format is **rowKey,f1:c1,f1:c2,f2:c1**. The number of columns must be the same as the number of attributes specified in the source stream.
timeindicator	No	Timestamp added in the source stream. The value can be **Processing Time** or **Event Time**. **NOTE** ● If this parameter is set to **Processing Time**, set **timeindicator** to **proctime.proctime**. In this case, an attribute proctime will be added to the original attribute field. If there are three attributes in the original attribute field, four attributes will be exported after this parameter is set to **processing time**. ● If this parameter is set to **Event Time**, you can select an attribute in the stream as the timestamp that is in the format of attr_name.rowtime, where attr_name indicates the attribute in the stream. ● This parameter can be simultaneously set to **processing time** and **event time**.

Precautions

The attribute type used as the timestamp must be long or timestamp.

Example

Read the **car_infos** table from HBase of CloudTable.

```
CREATE SOURCE STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT
)
WITH (
  type = "cloudtable",
  region = "cn-north-1" ,
```

```
  cluster_id = "209ab1b6-de25-4c48-8e1e-29e09d02de28",
  table_name = "carinfo",
  table_columns = "rowKey,info:owner,info:age,car:speed,car:miles"
);
```

5.2 HBase Sink Stream

Overview

CS exports the job output data to HBase of CloudTable. HBase is a column-oriented distributed cloud storage system that features enhanced reliability, excellent performance, and elastic scalability. It applies to the storage of massive amounts of data and distributed computing. You can use HBase to build a storage system capable of storing TB- or even PB-level data. With HBase, you can filter and analyze data with ease and get responses in milliseconds, rapidly mining data value. Structured and semi-structured key-value data can be stored, including messages, reports, recommendation data, risk control data, logs, and orders. With CS, you can write massive volumes of data to HBase at a high speed and with low latency.

CloudTable is a distributed, scalable, and fully-hosted key-value data storage service based on Apache HBase. It provides CS with high-performance random read and write capabilities, which are helpful when applications need to store and query a massive amount of structured data, semi-structured data, and time series data. CloudTable applies to IoT scenarios and storage and query of massive volumes of key-value data. For more information about CloudTable, see the **CloudTable Service User Guide**.

Syntax

Syntax

CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "cloudtable",region = "",cluster_id = "",table_name = "",table_columns = "",create_if_not_exist = "")

Description

Table 5-2 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **cloudtable** indicates that data is stored to CloudTable (HBase).
region	Yes	Region to which CloudTable belongs.
cluster_id	Yes	ID of the cluster to which the data table to be read belongs.
table_name	Yes	Name of the table, into which data is to be inserted. It can be specified through parameter configurations. For example, if you want one or more certain columns as part of the table name, use **car_pass_inspect_with_age_${car_age}**, where **car_age** is the column name.

Parameter	Mandatory	Description
table_columns	Yes	Columns to be inserted. The parameter value is the following format: **rowKey, f1:c1, f1:c2, f2:c1**, where **rowKey** must be specified. If you do not want to add a column, for example the third column, to the database, set this parameter to **rowKey,f1:c1,,f2:c1**.
illegal_data_table	No	If this parameter is specified, abnormal data (for example, **rowKey** does not exist) will be written into the table. If not specified, abnormal data will be discarded. The rowKey value is a timestamp followed by six random digits, and the schema is info:data, info:reason.
create_if_not_exist	No	Whether to create a table or column into which the data is written when this table or column does not exist. The value can be **true** or **false**, and **false** is used by default.
batch_insert_data_num	No	Number of data records to be written in batches at a time. The value must be a positive integer. The upper limit is **100**. The default value is **10**.

Precautions

If a configuration item can be specified through parameter configurations, one or more columns in the record can be used as part of the configuration item. For example, if the configuration item is set to **car_$ {car_brand}** and the value of **car_brand** in a record is **BMW**, the value of this configuration item is **car_BMW** in the record.

Example

Output data of stream **qualified_cars** to CloudTable (HBase).

```
CREATE SINK STREAM qualified_cars (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT
)
WITH (
  type = "cloudtable",
  region = "cn-north-1" ,
  cluster_id = "209ab1b6-de25-4c48-8e1e-29e09d02de28",
  table_name = "car_pass_inspect_with_age_${car_age}",
  table_columns = "rowKey,info:owner,,car:speed,car:miles",
  illegal_data_table = "illegal_data",
  create_if_not_exist = "true",
  batch_insert_data_num = "20"
);
```

5.3 OpenTSDB Sink Stream

Overview

CS exports the job output data to OpenTSDB of CloudTable. OpenTSDB is a distributed, scalable time series database based on HBase. It stores time series data. Time series data refers to the data collected at different time points. This type of data reflects the change status or degree of an object over time. OpenTSDB supports data collection and monitoring in seconds, permanent storage, index, and queries. It can be used for system monitoring and measurement as well as collection and monitoring of IoT data, financial data, and scientific experimental results.

CloudTable is a distributed, scalable, and fully-hosted key-value data storage service based on Apache HBase. It provides CS with high-performance random read and write capabilities, which are helpful when applications need to store and query a massive amount of structured data, semi-structured data, and time series data. CloudTable applies to IoT scenarios and storage and query of massive volumes of key-value data. For more information about CloudTable, see the **CloudTable Service User Guide**.

Prerequisites

Ensure that OpenTSDB has been enabled on the CloudTable clusters. For details about how to enable OpenTSDB, see **Enabling OpenTSDB** in the *CloudTable Service User Guide*.

Syntax

Syntax

CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "opentsdb",region = "",cluster_id = "",tsdb_metrics = "", tsdb_timestamps = "",tsdb_values = "",tsdb_tags = "",batch_insert_data_num = "")

Description

Table 5-3 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **opentsdb** indicates that data is stored to CloudTable (OpenTSDB).
region	Yes	Region to which CloudTable belongs.
cluster_id	Yes	ID of the cluster to which data is to be inserted.
tsdb_metrics	Yes	Metric of a data point, which can be specified through parameter configurations.
tsdb_timestamps	Yes	Timestamp of a data point. The data type can be TIMESTAMP, LONG, or INT. Only dynamic columns are supported.

Parameter	Mandatory	Description
tsdb_values	Yes	Value of a data point. The data type can be SHORT, INT, LONG, FLOAT, DOUBLE, or STRING. Dynamic columns or constant values are supported.
tsdb_tags	Yes	Tags of a data point. Each of tags contains at least one tag value and up to eight tag values. Tags of the data point can be specified through parameter configurations.
batch_insert_data_num	No	Number of data records to be written in batches at a time. The value must be a positive integer. The upper limit is **100**. The default value is **8**.

Precautions

If a configuration item can be specified through parameter configurations, one or more columns in the record can be used as part of the configuration item. For example, if the configuration item is set to **car_$ {car_brand}** and the value of **car_brand** in a record is **BMW**, the value of this configuration item is **car_BMW** in the record.

Example

Output data of stream **weather_out** to CloudTable (OpenTSDB).

```
CREATE SINK STREAM weather_out (
  timestamp_value LONG, /* Time */
  temperature FLOAT, /* Temperature value */
  humidity FLOAT, /* Humidity */
  location STRING /* Location */
)
WITH (
  type = "opentsdb",
  region = "cn-north-1" ,
  cluster_id = "e05649d6-00e2-44b4-b0ff-7194adaeab3f",
  tsdb_metrics = "weather",
  tsdb_timestamps = "${timestamp_value}",
  tsdb_values = "${temperature}; ${humidity}",
  tsdb_tags = "location:${location},signify:temperature; location:$
{location},signify:humidity",
  batch_insert_data_num = "10"
);
```


6 Cloud Ecosystem: RDS

6.1 RDS Sink Stream

Overview

CS outputs the job output data to RDS. Currently, PostgreSQL and MySQL databases are supported. The PostgreSQL database can store data of more complex types and delivers space information services, multi-version concurrent control (MVCC), and high concurrency. It applies to location applications, financial insurance, and e-commerce. The MySQL database reduces IT deployment and maintenance costs in various scenarios, such as web applications, e-commerce, enterprise applications, and mobile applications.

RDS is a cloud-based web service. RDS includes databases of the following types: MySQL, HWSQL, PostgreSQL, and Microsoft SQL Server. For more information about RDS, see the **Relational Database Service User Guide**.

Prerequisites

- Ensure that you have created a PostgreSQL or MySQL RDS instance in RDS.

  For details about how to create an RDS instance, see **Buying an Instance** in the **Relational Database Service Getting Started Guide**.
- In this scenario, jobs must run on the exclusive cluster of CS. Therefore, CS must interconnect with the VPC that has been connected with RDS instance. You can also set the security group rules as required.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

  For details about how to configure security group rules, see **Security Group** in the **Virtual Private Cloud User Guide**.

Syntax

Syntax

CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "rds",username = "",password = "",db_url = "",table_name = "");

Description

Table 6-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **rds** indicates that data is stored to RDS.
username	Yes	Username for connecting to a database.
password	Yes	Password for connecting to a database.
db_url	Yes	Database connection address, for example, **{database_type}://ip:port/database**. Currently, two types of database connections are supported: MySQL and PostgreSQL. ● MySQL: 'mysql://ip:port/database' ● PostgreSQL: 'postgresql://ip:port/database'
table_name	Yes	Name of the table where data will be inserted.
db_columns	No	Mapping between attributes in the output stream and those in the database table. This parameter must be configured based on the sequence of attributes in the output stream. Example: `create sink stream a3(student_name  string, student_age int) with ( type = "rds", username = "root", password = "xxxxxxxx", db_url = "mysql://192.168.0.102:8635/test1", db_columns = "name,age", table_name = "t1" );` In the example, **student_name** corresponds to the name attribute in the database, and **student_age** corresponds to the age attribute in the database. **NOTE** If **db_columns** is not configured, it is normal that the number of attributes in the output stream is less than that of attributes in the database table and the extra attributes in the database table are all nullable or have default values.

Parameter	Mandatory	Description
primary_key	No	To update data in the table in real time by using the primary key, add the **primary_key** configuration item (**c_timeminute** in the following example) when creating a table. During the data write operation, data is updated if the specified **primary_key** exists. Otherwise, data is inserted. Example: `CREATE SINK STREAM test(c_timeminute LONG, c_cnt LONG) WITH ( type = "rds", username = "root", password = "xxxxxxxx", db_url = "mysql://192.168.0.12:8635/test", table_name = "test", primary_key = "c_timeminute");`
operation_field	No	Processing method of specified data in the format of ${field_name}. The value of **field_name** must be a string. If **field_name** indicates D or DELETE, this record is deleted from the database and data is inserted by default.

Precautions

The stream format defined by **stream_id** must be the same as the table format.

Example

Data of stream **audi_cheaper_than_30w** is exported to the **audi_cheaper_than_30w** table in the **test** database.

```
CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
WITH (
  type = "rds",
  username = "root",
  password = "xxxxxx",
  db_url = "mysql://192.168.1.1:8635/test",
  table_name = "audi_cheaper_than_30w"
);
```


7 Cloud Ecosystem: DWS

7.1 DWS Sink Stream (JDBC Mode)

Overview

CS outputs the job output data to Data Warehouse Service (DWS). DWS database kernel is compliant with PostgreSQL. The PostgreSQL database can store data of more complex types and delivers space information services, multi-version concurrent control (MVCC), and high concurrency. It applies to location applications, financial insurance, and e-commerce.

DWS is an online data processing database based on the public cloud infrastructure and platform and helps you mine and analyze massive sets of data. For more information about DWS, see the **Data Warehouse Service Management Guide**.

Prerequisites

- Ensure that you have created a DWS cluster on DWS using your account.

  For details about how to create a DWS cluster, see **Creating a Cluster** in the *Data Warehouse Service Management Guide*.
- Ensure that a DWS database table has been created.
- In this scenario, jobs must run on the exclusive cluster of CS. Therefore, CS must interconnect with the VPC that has been connected with DWS clusters. You can also set the security group rules as required.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

  For details about how to configure security group rules, see **Security Group** in the **Virtual Private Cloud User Guide**.

Syntax

Syntax

CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "rds",username = "",password = "",db_url = "",table_name = "");

Description

Table 7-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **rds** indicates that data is stored to RDS or DWS.
username	Yes	Username for connecting to a database.
password	Yes	Password for connecting to a database.
db_url	Yes	Format of the address used to connect to the database, which is as follows: postgresql://ip:port/database
table_name	Yes	Name of the table where data will be inserted. You need to create the database table in advance.

db_columns	No	Mapping between attributes in the output stream and those in the database table. This parameter must be configured based on the sequence of attributes in the output stream.
		Example:
		`create sink stream a3(student_name  string, student_age int) with (`
		`type = "rds",`
		`username = "root",`
		`password = "xxxxxxxx",`
		`db_url = "postgresql://192.168.0.102:8635/test1",`
		`db_columns = "name,age",`
		`table_name = "t1"`
		`);`
		In the example, **student_name** corresponds to the name attribute in the database, and **student_age** corresponds to the age attribute in the database.
		NOTE
		If **db_columns** is not configured, it is normal that the number of attributes in the output stream is less than that of attributes in the database table and the extra attributes in the database table are all nullable or have default values.

primary_key	No	To update data in the table in real time by using the primary key, add the **primary_key** configuration item (**c_timeminute** in the following example) when creating a table. During the data write operation, data is updated if the specified **primary_key** exists. Otherwise, data is inserted.
		Example:
		`CREATE SINK STREAM test(c_timeminute LONG, c_cnt LONG)`
		`WITH (`
		`type = "rds",`
		`username = "root",`
		`password = "xxxxxxxx",`
		`db_url = "postgresql://192.168.0.12:8635/test",`
		`table_name = "test",`
		`primary_key = "c_timeminute");`

Precautions

The stream format defined by **stream_id** must be the same as the table format.

Example

Data of stream **audi_cheaper_than_30w** is exported to the **audi_cheaper_than_30w** table in the **test** database.

```
CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
WITH (
  type = "rds",
  username = "root",
  password = "xxxxxx",
  db_url = "postgresql://192.168.1.1:8635/test",
  table_name = "audi_cheaper_than_30w"
);
```

7.2 DWS Sink Stream (OBS-based Dumping)

Overview

Create a sink stream to export CS data to DWS through OBS-based dumping, specifically, output CS data to OBS and then import data from OBS to DWS. For details about how to import OBS data to DWS, see **Concurrently Importing Data from OBS** in the **Data Warehouse Service Development Guide**.

DWS is an online data processing database based on the public cloud infrastructure and platform and helps you mine and analyze massive sets of data. For more information about DWS, see the **Data Warehouse Service Management Guide**.

Precautions

- OBS-based dumping supports intermediate files of the following two types:
  - ORC: The ORC format does not support the array data type. If the ORC format is used, create a foreign server in DWS. For details, see **Creating a Foreign Server** in the **Data Warehouse Service Database Development Guide**.
  - CSV: By default, the line break is used as the record separator. If the line break is contained in the attribute content, you are advised to configure quote. For details, see **Table 7-2**.
- If the target table does not exist, a table is automatically created. CS data of the SQL type does not support text. If a long text exists, you are advised to create a table in the database.
- Ensure that OBS buckets and folders have been created.

  For details about how to create an OBS bucket, see **Creating a Bucket** in the **Object Storage Service Console Operation Guide**.

  For details about how to create a folder, see **Creating a Folder** in the **Object Storage Service Console Operation Guide**.

Syntax

Syntax

```
CREATE SINK STREAM stream_id (
   attr_name attr_type (',' attr_name attr_type)*
)WITH (
          type = "dws",
          region = "",
          encode = "",
          field_delimiter = "",
          quote = "",
          db_obs_server = "",
          obs_dir = "",
          username = "",
          password =  "",
          db_url = "",
          table_name = "",
          max_record_num_per_file = "",
          max_dump_file_num = "",
          dump_interval = ""
);
```

Description

Table 7-2 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **dws** indicates that data is stored to DWS.
region	Yes	Region where DWS is located.
encode	Yes	Encoding format. Currently, CSV and ORC are supported.
field_delimiter	No	Separator used to separate every two attributes. This parameter needs to be configured if the CSV encoding mode is used. It is recommended that you use invisible characters as separators, for example, **\u0006\u0002**.
quote	No	Single byte. It is recommended that invisible characters be used, for example, **u0007**.
db_obs_server	No	Foreign server (for example, **obs_server**) that has been created in the database. For details about how to create a foreign server, see **Creating a Foreign Server** in the *Data Warehouse Service Database Development Guide*. You need to specify this parameter if the ORC encoding mode is adopted.
obs_dir	Yes	Directory for storing intermediate files. The directory is in the format of {Bucket name}/{Directory name}, for example, obs-a1/dir1/subdir.

Parameter	Mandatory	Description
username	Yes	Username for connecting to a database.
password	Yes	Password for connecting to a database.
db_url	Yes	Database connection address. The format is /ip:port/database, for example, **192.168.1.21:8000/test1**.
table_name	Yes	Data table name. If no table is available, a table is automatically created.
max_record_num_per_file	Yes	Maximum number of records that can be stored in a file. If the number of records in a file is less than the maximum value, the file will be dumped to OBS after one dumping period.
max_dump_file_num	Yes	Maximum number of files that can be dumped at a time. If the number of files to be dumped is less than the maximum value, the files will be dumped to OBS after one dumping period.
dump_interval	Yes	Dumping period. The unit is second.

Example

- Dump files in CSV format.

```
CREATE SINK STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
WITH (
  type = "dws",
  region = "cn-north-1" ,
  encode = "csv",
  field_delimiter = "\u0006\u0006\u0002",
  quote = "\u0007",
  obs_dir = "cs-append-2/dws",
  username = "",
  password =  "",
  db_url = "192.168.1.12:8000/test1",
  table_name = "table1",
  max_record_num_per_file = "100",
  max_dump_file_num = "10",
  dump_interval = "10"
);
```

- Dump files in ORC format.

```
CREATE SINK STREAM car_infos (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT,
  car_timestamp LONG
)
WITH (
  type = "dws",
  region = "cn-north-1" ,
```



```
  encode = "orc",
  db_obs_server = "obs_server",
  obs_dir = "cs-append-2/dws",
  username = "",
  password =  "",
  db_url = "192.168.1.12:8000/test1",
  table_name = "table1",
  max_record_num_per_file = "100",
  max_dump_file_num = "10",
  dump_interval = "10"
);
```


8 Cloud Ecosystem: DDS

8.1 DDS Sink Stream

Overview

CS outputs the job output data to Document Database Service (DDS).

DDS is compatible with the MongoDB protocol and is secure, highly available, reliable, scalable, and easy to use. It provides DB instance creation, scaling, redundancy, backup, restoration, monitoring, and alarm reporting functions with just a few clicks on the DDS console. For more information about DDS, see the **Document Database Service User Guide**.

Prerequisites

- Ensure that you have created a DDS instance on DDS using your account.

  For details about how to create a DDS instance, see **Buying a DDS DB Instance** in the **Document Database Service Quick Start**.
- Currently, only instances that are not enabled with SSL authentication are supported.

Syntax

Syntax

CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "dds",username = "",password = "",db_url = "",field_names = "");

Description

Table 8-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **dds** indicates that data is stored to DDS.
username	Yes	Username for connecting to a database.

Parameter	Mandatory	Description
password	Yes	Password for connecting to a database.
db_url	Yes	DDS instance access address, for example, **ip1:port,ip2:port/database/collection**.
field_names	Yes	Key of the data field to be inserted. The specific format is as follows: "f1,f2,f3". Ensure that there is a one-to-one mapping between data in the parameter value and data columns in the sink stream.
batch_insert_data_num	No	Amount of data to be written in batches at a time. The value must be a positive integer. The default value is **10**.

Example

Output data in the **qualified_cars** stream to the **collectionTest** DDS DB.

```
CREATE SINK STREAM qualified_cars (
car_id STRING,
car_owner STRING,
car_age INT,
average_speed INT,
total_miles INT
)
WITH (
type = "dds",
region = "cn-north-1" ,
db_url = "192.168.0.8:8635,192.168.0.130:8635/dbtest/collectionTest",
username = "xxxxxxxxxx",
password =  "xxxxxxxxxx",
field_names = "car_id,car_owner,car_age,average_speed,total_miles",
batch_insert_data_num = "10"
);
```


9 Cloud Ecosystem: SMN

9.1 SMN Sink Stream

Overview

CS exports job output data to SMN.

SMN provides reliable and flexible large-scale message notification services to CS. It significantly simplifies system coupling and pushes messages to subscription endpoints based on requirements. SMN can be connected to other cloud services or integrated with any application that uses or generates message notifications to push messages over multiple protocols. For more information about SMN, see the **Simple Message Notification User Guide**.

Syntax

Syntax

CREATE SINK STREAM stream_id xxx WITH(type = "smn",region = "",topic_urn = "",urn_column = "",message_subject = "",message_column = "")

Description

Table 9-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **smn** indicates that data is stored to SMN.
region	Yes	Region to which SMN belongs.

Parameter	Mandatory	Description
topic_urn	No	URN of an SMN topic, which is used for the static topic URN configuration. The SMN topic serves as the destination for short message notification and needs to be created in SMN. One of **topic_urn** and **urn_column** must be configured. If both of them are configured, the **topic_urn** setting takes precedence.
urn_column	No	Field name containing the topic URN content, which is used for the dynamic topic URN configuration. One of **topic_urn** and **urn_column** must be configured. If both of them are configured, the **topic_urn** setting takes precedence.
message_subject	Yes	Message subject sent to SMN. This parameter can be user-defined.
message_column	Yes	Field name in the sink stream. Contents of the field name serve as the message contents, which are user-defined. Currently, only text messages (default) are supported.

Precautions

None

Example

Data of stream **over_speed_warning** is exported to SMN.

```
//Static topic configuration
CREATE SINK STREAM over_speed_warning (
  over_speed_message STRING /* over speed message */
)
WITH (
  type = "smn",
  region = "cn-north-1" ,
  topic_Urn = "urn:smn:cn-north-1:38834633fd6f4bae813031b5985dbdea:ddd",
  message_subject = "message title",
  message_column = "over_speed_message"
);
//Dynamic topic configuration
CREATE SINK STREAM over_speed_warning2 (
    over_speed_message STRING, /* over speed message */
    over_speed_urn STRING
)
WITH (
type = "smn",
region = "cn-north-1" ,
urn_column = "over_speed_urn",
```

```
message_subject = "message title",
message_column = "over_speed_message"
);
```


10 Cloud Ecosystem: Cloud Search Service

10.1 Elasticsearch Sink Stream

Overview

CS exports job output data to Elasticsearch of Cloud Search Service (CSS). Elasticsearch is a popular enterprise-class Lucene-powered search server and provides the distributed multi-user capabilities. It delivers multiple functions, including full-text retrieval, structured search, analytics, aggregation, and highlighting. With Elasticsearch, you can achieve stable, reliable, real-time search. Elasticsearch applies to diversified scenarios, such as log analysis and site search.

CSS is a fully managed, distributed search service. It is fully compatible with open-source Elasticsearch and provides CS with structured and unstructured data search, statistics, and report capabilities. For more information about CSS, see the **Cloud Search Service User Guide**.

Prerequisites

- Ensure that you have created a cluster on CSS using your account. For details about how to create a cluster on CSS, see **Creating a Cluster** in the *Cloud Search Service User Guide*.
- In this scenario, jobs must run on the exclusive cluster of CS. Therefore, CS must interconnect with the VPC that has been connected with CSS. You can also set the security group rules as required.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

  For details about how to configure security group rules, see **Security Group** in the **Virtual Private Cloud User Guide**.

Syntax

Syntax

CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "es",region = "",cluster_address = "",es_index = "",es_type= "",es_fields= "",batch_insert_data_num= "");

Description

Table 10-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **es** indicates that data is stored to CSS.
region	Yes	Region where CSS is located. For example, **cn-north-1**.
cluster_address	Yes	Private access address of the CSS cluster, for example: **x.x.x.x:x**. Use commas (,) to separate multiple addresses.
es_index	Yes	Index storing the data to be inserted.
es_type	Yes	Document type of the data to be inserted.
es_fields	Yes	Key of the data field to be inserted. The parameter is in the format of "Id, f1, f2, f3, f4". Ensure that the parameter value has a one-to-one mapping with data columns in the sink stream. If the key is not used, remove the **id** keyword. Specifically, the parameter is in the format of "F1, f2, f3, f4, f5".
batch_insert_data_num	Yes	Amount of data to be written in batches at a time. The value must be a positive integer. The upper limit is **100**. The default value is **10**.

Precautions

None

Example

Data of stream **qualified_cars** is exported to the cluster on CSS.

```
CREATE SINK STREAM qualified_cars (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT
)
WITH (
  type = "es",
  region = "cn-north-1" ,
  cluster_address = "192.168.0.212:9200",
  es_index = "china",
  es_type = "zhejiang",
  es_fields = "id,owner,age,speed,miles",
  batch_insert_data_num = "10"
);
```

11 Cloud Ecosystem: DCS

11.1 DCS Sink Stream

Overview

CS exports job output data to Redis of DCS. Redis is a storage system that supports multiple types of data structures such as key-value. It can be used in scenarios such as caching, event pub/sub, and high-speed queuing. Redis supports direct read/write of strings, hashes, lists, queues, and sets. Redis works with in-memory dataset and provides persistence. For more information about Redis, visit **https://redis.io/**.

DCS provides Redis-compatible, secure, reliable, out-of-the-box, distributed cache capabilities allowing elastic scaling and convenient management. It meets users' requirements for high concurrency and fast data access. For more information about DCS, see the **Distributed Cache Service User Guide**.

Prerequisites

- Ensure that You have created a Redis cache instance on DCS using your account.

  For details about how to create a Redis cache instance, see **Creating a DCS Instance** in the **Distributed Cache Service User Guide**.
- In this scenario, jobs must run on the exclusive cluster of CS. Therefore, CS must interconnect with the VPC that has been connected with DCS clusters. You can also set the security group rules as required.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

  For details about how to configure security group rules, see **Security Group** in the **Virtual Private Cloud User Guide**.
- If you use a VPC peering connection to access a DCS instance, the following restrictions also apply:
  - If network segment 172.16.0.0/12~24 is used during DCS instance creation, the CS cluster cannot be in any of the following network segments: 192.168.1.0/24, 192.168.2.0/24, and 192.168.3.0/24.

- If network segment 192.168.0.0/16~24 is used during DCS instance creation, the CS cluster cannot be in any of the following network segments: 172.31.1.0/24, 172.31.2.0/24, and 172.31.3.0/24.
- If network segment 10.0.0.0/8~24 is used during DCS instance creation, the CS cluster cannot be in any of the following network segments: 172.31.1.0/24, 172.31.2.0/24, and 172.31.3.0/24.

Syntax

Syntax

CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "dcs_redis",region = "",cluster_address = "",password = "",value_type= "",key_value= "");

Description

Table 11-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **dcs_redis** indicates that data is exported to DCS Redis.
region	Yes	Region where DCS for storing the data is located.
cluster_address	Yes	Redis instance connection address.
password	No	Redis instance connection password. This parameter is not required if password-free access is used.
value_type	Yes	This parameter can be set to any or the combination of the following options: ● Data types, including **string**, **list**, **hash**, **set**, and **zset** ● Commands used to set the expiration time of a key, including **expire**, **pexpire**, **expireAt**, and **pexpireAt** ● Commands used to delete a key, including **del** and **hdel** Use commas (,) to separate multiple commands.

Parameter	Mandatory	Description
key_value	Yes	Key and value. The number of key_value pairs must be the same as the number of types specified by value_type, and key_value pairs are separated by semicolons (;). Both key and value can be specified through parameter configurations. The dynamic column name is represented by ${column name}.

Precautions

- If a configuration item can be specified through parameter configurations, one or more columns in the record can be used as part of the configuration item. For example, if the configuration item is set to **car_$ {car_brand}** and the value of **car_brand** in a record is **BMW**, the value of this configuration item is **car_BMW** in the record.
- Characters ":", ",", ";", "$", "{", and "}" have been used as special separators without the escape function. These characters cannot be used in key and value as common characters. Otherwise, parsing will be affected and the program exceptions will occur.

Example

Data of stream **qualified_cars** is exported to the Redis cache instance on DCS.

```
CREATE SINK STREAM qualified_cars (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed DOUBLE,
  total_miles DOUBLE
)
WITH (
  type = "dcs_redis",
  cluster_address = "192.168.0.34:6379",
  password = "xxxxxxxx",
  value_type = "string; list; hash; set; zset",
  key_value = "${car_id}_str: ${car_owner}; name_list: ${car_owner}; $
{car_id}_hash: {name:${car_owner}, age: ${car_age}}; name_set:   ${car_owner};
math_zset: {${car_owner}:${average_speed}}"
);
```

12 Cloud Ecosystem: MRS

12.1 Kafka Source Stream

Overview

Create a source stream to obtain data from Kafka as input data for jobs.

Apache Kafka is a fast, scalable, and fault-tolerant distributed message publishing and subscription system. It delivers high throughput and built-in partitions and provides data replicas and fault tolerance. Apache Kafka is applicable to scenarios of handling massive messages. Kafka clusters are deployed and hosted on MRS that is powered on Apache Kafka.

Prerequisites

- If the Kafka server listens on the port using hostname, you need to add the mapping between the hostname and IP address of the Kafka Broker node to the CS cluster. Contact the Kafka service deployment personnel to obtain the hostname and IP address of the Kafka Broker node. For details about how to add an IP-domain mapping, see the description of **Adding an IP-Domain Mapping** in **Cluster Management** in the *Cloud Stream Service User Guide*.
- When using offline Kafka clusters, use VPC peering connections to connect CS to Kafka.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

Syntax

Syntax

CREATE SOURCE STREAM kafka_source (name STRING, age int)WITH (type = "kafka",kafka_bootstrap_servers = "",kafka_group_id = "",kafka_topic = "",encode = "json")(TIMESTAMP BY timeindicator (',' timeindicator)?);timeindicator:PROCTIME '.' PROCTIME| ID '.' ROWTIME

Description

Table 12-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Data source type. Value **Kafka** indicates that the data source is Kafka.
kafka_bootstrap_servers	Yes	Port that connects CS to Kafka. Use VPC peering connections to connect CS clusters with Kafka clusters.
kafka_group_id	Yes	Group ID.
kafka_topic	Yes	Kafka topic to be read.
encode	Yes	Data encoding format. The value can be **csv**, **json**, or **blob**. ● **field_delimiter** must be specified if this parameter is set to **csv**. ● **json_config** must be specified if this parameter is set to **json**. ● If this parameter is set to **blob**, the received data is not parsed, only one stream attribute exists, and the stream attribute is of the Array[TINYINT] type.
json_config	No	If **encode** is set to **json**, you can use this parameter to specify the mapping between JSON fields and stream attribute fields. The format is field1=json_field1;field2=json_field2.
field_delimiter	No	If **encode** is set to **csv**, you can use this parameter to specify the separator between CSV fields. By default, the comma (,) is used.
quote	No	Quoted symbol in a data format. The attribute delimiters between two quoted symbols are treated as common characters. ● If double quotation marks are used as the quoted symbol, set this parameter to "\u005c \u0022" for character conversion. ● If a single quotation mark is used as the quoted symbol, set this parameter to a comma (,). **NOTE** After this parameter is specified, ensure that each field does not contain quoted symbols or contains an even number of quoted symbols. Otherwise, parsing will fail.

Parameter	Mandatory	Description
timeindicator	No	Timestamp added in the source stream. The value can be **processing time** or **event time**. **NOTE** ● If this parameter is set to **processing time**, the format is **proctime.proctime**. In this case, an attribute proctime will be added to the original attribute field. If there are three attributes in the original attribute field, four attributes will be exported after this parameter is set to **processing time**. However, the attribute length remains unchanged if the rowtime attribute is specified. ● If this parameter is set to **event time**, you can select an attribute in the stream as the timestamp. The format is **attr_name.rowtime**. ● This parameter can be simultaneously set to **processing time** and **event time**.
start_time	No	Start time when Kafka data is ingested. If this parameter is specified, CS reads data read from the specified time. The parameter value is in the format of yyyy-MM-dd HH:mm:ss. Ensure that the value of **start_time** is not later than the current time. Otherwise, no data will be obtained.

Precautions

The attribute type used as the timestamp must be long or timestamp.

Example

Read data from the Kafka topic **test**.

```
CREATE SOURCE STREAM kafka_source (name STRING, age int)
WITH (
  type = "kafka",
  kafka_bootstrap_servers = "ip1:port1,ip2:port2",
  kafka_group_id = "sourcegroup1",
  kafka_topic = "test",
  encode = "json"
);
```

12.2 Kafka Sink Stream

Overview

CS exports the job output data to Kafka.

Apache Kafka is a fast, scalable, and fault-tolerant distributed message publishing and subscription system. It delivers high throughput and built-in partitions and provides data

replicas and fault tolerance. Apache Kafka is applicable to scenarios of handling massive messages. Kafka clusters are deployed and hosted on MRS that is powered on Apache Kafka.

Prerequisites

- If the Kafka server listens on the port using hostname, you need to add the mapping between the hostname and IP address of the Kafka Broker node to the CS cluster. Contact the Kafka service deployment personnel to obtain the hostname and IP address of the Kafka Broker node. For details about how to add an IP-domain mapping, see the description of **Adding an IP-Domain Mapping** in **Cluster Management** in the *Cloud Stream Service User Guide*.
- When using offline Kafka clusters, use VPC peering connections to connect CS to Kafka.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

Syntax

Syntax

CREATE SINK STREAM kafka_sink (name STRING) WITH(type = "kafka",kafka_bootstrap_servers = "",kafka_topic = "",encode = "json")

Description

Table 12-2 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **kafka** indicates that data is stored to Kafka.
kafka_bootstrap_servers	Yes	Port that connects CS to Kafka. Use VPC peering connections to connect CS clusters with Kafka clusters.
kafka_topic	Yes	Kafka topic into which CS writes data.
encode	Yes	Encoding format. Currently, only JSON is supported.

Precautions

None

Example

Output data to Kafka.

```
CREATE SINK STREAM kafka_sink (name STRING)
WITH (
  type="kafka",
  kafka_bootstrap_servers =  "ip1:port1,ip2:port2",
  kafka_topic = "testsink",
```



```
  encode = "json"
);
```

12.3 HBase Sink Stream

Overview

CS exports the job output data to HBase of MRS.

Prerequisites

- An MRS cluster has been created by using your account. Currently, CS can only interconnect with the MRS cluster disabled with Kerberos authentication.
- In this scenario, the job needs to run in an exclusive cluster. Ensure that an exclusive cluster has been created.

  For details about how to create an exclusive cluster, see **Cluster Management** in the *Cloud Stream Service User Guide*.
- Ensure that a VPC peering connection has been set up between the exclusive cluster and the MRS cluster, and security group rules have been configured based on the site requirements.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

  For details about how to configure security group rules, see **Security Group** in the **Virtual Private Cloud User Guide**.

Syntax

Syntax

CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "mrs_hbase",region = "",cluster_address = "",table_name = "",table_columns = "",illegal_data_table = "",batch_insert_data_num = "",action = "")

Description

Table 12-3 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **mrs_hbase** indicates that data is stored to HBase of MRS.
region	Yes	Region where MRS resides.
cluster_address	Yes	ZooKeeper address of the cluster to which the table where data is to be inserted belongs. The value is in the format of ip1,ip2:port.

Parameter	Mandatory	Description
table_name	Yes	Name of the table where data is to be inserted. It can be specified through parameter configurations. For example, if you want one or more certain columns as part of the table name, use **car_pass_inspect_with_age_${car_age}**, where **car_age** is the column name.
table_columns	Yes	Columns to be inserted. The parameter value is the following format: **rowKey, f1:c1, f1:c2, f2:c1**, where **rowKey** must be specified. If you do not want to add a column, for example the third column, to the database, set this parameter to **rowKey,f1:c1,,f2:c1**.
illegal_data_table	No	If this parameter is specified, abnormal data (for example, **rowKey** does not exist) will be written into the table. If not specified, abnormal data will be discarded. The rowKey value is taskNo_Timestamp followed by six random digits, and the schema is info:data, info:reason.
batch_insert_data_num	No	Number of data records to be written in batches at a time. The value must be a positive integer. The upper limit is **1000**. The default value is **10**.
action	No	Whether data is added or deleted. Available options include **add** and **delete**. The default value is **add**.

Precautions

None

Example

Output data to HBase of MRS.

```
CREATE SINK STREAM qualified_cars (
  car_id STRING,
  car_owner STRING,
  car_age INT,
  average_speed INT,
  total_miles INT
)
WITH (
  type = "mrs_hbase",
  region = "cn-north-1" ,
  cluster_address = "192.16.0.88,192.87.3.88:2181",
  table_name = "car_pass_inspect_with_age_${car_age}",
  table_columns = "rowKey,info:owner,,car:speed,car:miles",
  illegal_data_table = "illegal_data",
  batch_insert_data_num = "20",
  action = "add"
);
```

12.4 Interaction with Kafka Using User-Defined Jobs

Overview

You can perform secondary development based on Flink and Spark APIs to build your own JAR packages and submit them to the CS cluster to implement interactions with CS and MRS Kafka clusters.

Apache Kafka is a fast, scalable, and fault-tolerant distributed message publishing and subscription system. It delivers high throughput and built-in partitions and provides data replicas and fault tolerance. Apache Kafka is applicable to scenarios of handling massive messages. Kafka clusters are deployed and hosted on MRS that is powered on Apache Kafka.

Prerequisites

- To use Kafka in an MRS cluster, you need to use the VPC peering connection to interconnect CS with the MRS cluster.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

- If the Kafka server listens on the port using hostname, you need to add the mapping between the hostname and IP address of the Kafka Broker node to the CS cluster. Contact the Kafka service deployment personnel to obtain the hostname and IP address of the Kafka Broker node. For details about how to add an IP-domain mapping, see the description of **Adding an IP-Domain Mapping** in **Cluster Management** in the *Cloud Stream Service User Guide*.

Procedure

Create and submit a user-defined Flink job. For details, see **Creating a User-Defined Flink Job** in the *Cloud Stream Service User Guide*.

Create and submit a user-defined Spark job. For details, see **Creating a User-Defined Spark Job** in the *Cloud Stream Service User Guide*.

12.5 Interaction with HBase Using User-Defined Jobs

Overview

You can perform secondary development based on Flink and Spark APIs to build your own JAR packages and submit them to the CS cluster to implement interactions with CS and MRS HBase clusters.

Apache HBase is a column-oriented distributed cloud storage system that features enhanced reliability, excellent performance, and elastic scalability. It applies to the storage of massive amounts of data and distributed computing. You can use HBase to build a storage system capable of storing TB- or even PB-level data. With HBase, you can filter and analyze data with ease and get responses in milliseconds, rapidly mining data value. HBase clusters are deployed and hosted on MRS that is powered on Apache HBase.

Prerequisites

- To use Kafka in an MRS cluster, you need to use the VPC peering connection to interconnect CS with the MRS cluster.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

- If the Kafka server listens on the port using hostname, you need to add the mapping between the hostname and IP address of the Kafka Broker node to the CS cluster. Contact the Kafka service deployment personnel to obtain the hostname and IP address of the Kafka Broker node. For details about how to add an IP-domain mapping, see the description of **Adding an IP-Domain Mapping** in **Cluster Management** in the *Cloud Stream Service User Guide*.

Procedure

Create and submit a user-defined Flink job. For details, see **Creating a User-Defined Flink Job** in the *Cloud Stream Service User Guide*.

Create and submit a user-defined Spark job. For details, see **Creating a User-Defined Spark Job** in the *Cloud Stream Service User Guide*.

13 Cloud Ecosystem: APIG

13.1 APIG Sink Stream

Overview

CS outputs the job result to open APIs of APIG. You can access data through API calling on APIG. APIG is an API hosting service with high performance, availability, and security. It allows users to create, manage, and deploy APIs at any scale. With APIG, you can implement system integration, micro-service aggregation, and serverless architectures easily and quickly with low costs and risks. For more information about API Gateway, see the **API Gateway User Guide (API Calling)**.

Prerequisites

- Ensure that you have created an application on APIG using your account. For details about how to create an application using API calling, see **Creating an Application** in the **API Gateway User Guide (API Calling)**.

Syntax

Syntax

CREATE SINK STREAM stream_id (attr_name attr_type (',' attr_name attr_type)*)WITH (type = "apig",region = "",app_id= "",encode= "",field_delimiter= "");

Description

Table 13-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **apig** indicates that data is stored to APIG.
region	Yes	Region where APIG for storing the data is located.

Parameter	Mandatory	Description
app_id	Yes	ID of the application used for API calling.
encode	Yes	Data encoding format. The value can be **csv** and **json**. **NOTE** ● If the encoding format is **csv**, you need to configure field separators. ● If the encoding format is **json**, you need to configure whether to generate an empty field. For details, see the examples.
field_delimiter	Yes	Separator used to separate every two attributes. ● This parameter needs to be configured if the CSV encoding format is adopted. It can be user-defined, for example, a comma (**,**). ● This parameter is not required if the JSON encoding format is adopted.
json_config	No	If **encode** is set to **json**, you can set this parameter to specify the mapping between the JSON field and the stream definition field. An example of the format is as follows: field1=data_json.field1; field2=data_json.field2.

Precautions

None

Example

- CSV: Data is written to APIG as codes in CSV format which are separated by commas (,). If there are multiple partitions, car_owner is used as the key to distribute data to different partitions. An example is as follows: "ZJA710XC", "lilei", "BMW", 700000.

```
CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
WITH (
  type = "apig",
  app_id = "xxxxxxxxxx",
  region = "cn-north-1" ,
  encode = "csv",
  field_delimiter = ","
);
```


- JSON: Data is output to APIG and encoded using JSON. If there are multiple partitions, car_owner and car_brand are used as the keys to distribute data to different partitions. If **enableOutputNull** is set to **true**, an empty field (the value is **null**) is generated. If set to **false**, no empty field is generated. An example is as follows: "car_id ":"ZJA710XC", "car_owner ":"lilei", "car_brand ":"BMW", "car_price ":700000.

```
CREATE SINK STREAM audi_cheaper_than_30w (
  car_id STRING,
  car_owner STRING,
  car_brand STRING,
  car_price INT
)
WITH (
  type = "apig",
  app_id = "6ac32a6596614bf69fb5c5553f26963f",
  region = "cn-north-1" ,
  encode = "json",
  enable_output_null = "false"
);
```


14 Cloud Ecosystem: DMS

Distributed Message Service (DMS) is a message middleware service based on distributed, high-availability clustering technology. It provides reliable, scalable, fully managed queues for sending, receiving, and storing messages. DMS for Kafka is compatible with open-source Kafka. It provides you with fully-managed, highly reliable Kafka queues and Kafka premium instances with exclusive computing, storage, and bandwidth resources.

For more information about the Kafka premium instances, see **Kafka Premium Instance Management** in the **Distributed Message Service User Guide**.

The source stream can read data from a Kafka premium instance as the input data of jobs. The syntax for creating a Kafka premium source stream is the same as that for creating an open source Apache Kafka source stream. For details, see **Kafka Source Stream**.

CS can also write the job output data into the Kafka premium instance. The syntax for creating a Kafka premium sink stream is the same as that for creating an open-source Apache Kafka sink stream. For details, see **Kafka Sink Stream**.

15 Open-source Ecosystem: Apache Kafka

15.1 Kafka Source Stream

Overview

Create a source stream to obtain data from Kafka as input data for jobs.

Apache Kafka is a fast, scalable, and fault-tolerant distributed message publishing and subscription system. It delivers high throughput and built-in partitions and provides data replicas and fault tolerance. Apache Kafka is applicable to scenarios of handling massive messages.

Prerequisites

- If the Kafka server listens on the port using hostname, you need to add the mapping between the hostname and IP address of the Kafka Broker node to the CS cluster. Contact the Kafka service deployment personnel to obtain the hostname and IP address of the Kafka Broker node. For details about how to add an IP-domain mapping, see the description of **Adding an IP-Domain Mapping** in **Cluster Management** in the *Cloud Stream Service User Guide*.
- When using offline Kafka clusters, use VPC peering connections to connect CS to Kafka.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

Syntax

Syntax

CREATE SOURCE STREAM kafka_source (name STRING, age int)WITH (type = "kafka",kafka_bootstrap_servers = "",kafka_group_id = "",kafka_topic = "",encode = "json",json_config="")(TIMESTAMP BY timeindicator (',' timeindicator)?);timeindicator:PROCTIME '.' PROCTIME| ID '.' ROWTIME

Description

Table 15-1 Syntax description

Parameter	Mandatory	Description
type	Yes	Data source type. Value **Kafka** indicates that the data source is Kafka.
kafka_bootstrap_servers	Yes	Port that connects CS to Kafka. Use VPC peering connections to connect CS clusters with Kafka clusters.
kafka_group_id	Yes	Group ID.
kafka_topic	Yes	Kafka topic to be read.
encode	Yes	Data encoding format. The value can be **json**, **csv**, or **blob**. ● **field_delimiter** must be specified if this parameter is set to **csv**. ● **json_config** must be specified if this parameter is set to **json**. ● If this parameter is set to **blob**, the received data is not parsed, only one stream attribute exists, and the stream attribute is of the Array[TINYINT] type.
json_config	No	If **encode** is set to **json**, you can use this parameter to specify the mapping between JSON fields and stream attribute fields. The format is field1=json_field1;field2=json_field2.
field_delimiter	No	If **encode** is set to **csv**, you can use this parameter to specify the separator between CSV fields. By default, the comma (,) is used.
quote	No	Quoted symbol in a data format. The attribute delimiters between two quoted symbols are treated as common characters. ● If double quotation marks are used as the quoted symbol, set this parameter to "\u005c\u0022" for character conversion. ● If a single quotation mark is used as the quoted symbol, set this parameter to a comma (,). **NOTE** After this parameter is specified, ensure that each field does not contain quoted symbols or contains an even number of quoted symbols. Otherwise, parsing will fail.

Parameter	Mandatory	Description
timeindicator	No	Timestamp added in the source stream. The value can be **processing time** or **event time**. **NOTE** ● If this parameter is set to **processing time**, the format is **proctime.proctime**. In this case, an attribute proctime will be added to the original attribute field. If there are three attributes in the original attribute field, four attributes will be exported after this parameter is set to **processing time**. However, the attribute length remains unchanged if the rowtime attribute is specified. ● If this parameter is set to **event time**, you can select an attribute in the stream as the timestamp. The format is **attr_name.rowtime**. ● This parameter can be simultaneously set to **processing time** and **event time**.
start_time	No	Start time when Kafka data is ingested. If this parameter is specified, CS reads data read from the specified time. The parameter value is in the format of yyyy-MM-dd HH:mm:ss. Ensure that the value of **start_time** is not later than the current time. Otherwise, no data will be obtained.

Precautions

The attribute type used as the timestamp must be long or timestamp.

Example

Read Kafka topic **test**. Data instance: {"attr1": "lilei", "attr2": 18}

```
CREATE SOURCE STREAM kafka_source (name STRING, age int)
WITH (
  type = "kafka",
  kafka_bootstrap_servers = "ip1:port1,ip2:port2",
  kafka_group_id = "sourcegroup1",
  kafka_topic = "test",
  encode = "json",
  json_config = "name=attr1;age=attr2"
);
```


15.2 Kafka Sink Stream

Overview

CS exports the job output data to Kafka.

Apache Kafka is a fast, scalable, and fault-tolerant distributed message publishing and subscription system. It delivers high throughput and built-in partitions and provides data replicas and fault tolerance. Apache Kafka is applicable to scenarios of handling massive messages.

Prerequisites

- If the Kafka server listens on the port using hostname, you need to add the mapping between the hostname and IP address of the Kafka Broker node to the CS cluster. Contact the Kafka service deployment personnel to obtain the hostname and IP address of the Kafka Broker node. For details about how to add an IP-domain mapping, see the description of **Adding an IP-Domain Mapping** in **Cluster Management** in the *Cloud Stream Service User Guide*.
- When using offline Kafka clusters, use VPC peering connections to connect CS to Kafka.

  For details about how to set up the VPC peering connection, see **VPC Peering Connection** in the *Cloud Stream Service User Guide*.

Syntax

Syntax

CREATE SINK STREAM kafka_sink (name STRING) WITH(type = "kafka",kafka_bootstrap_servers = "",kafka_topic = "",encode = "json")

Description

Table 15-2 Syntax description

Parameter	Mandatory	Description
type	Yes	Output channel type. Value **kafka** indicates that data is stored to Kafka.
kafka_bootstrap_servers	Yes	Port that connects CS to Kafka. Use VPC peering connections to connect CS clusters with Kafka clusters.
kafka_topic	Yes	Kafka topic into which CS writes data.
encode	Yes	This parameter can be set to **json** or **csv**.
filed_delimiter	No	If **encode** is set to **csv**, you can use this parameter to specify the separator between CSV fields. By default, the comma (,) is used.

Precautions

None

Example

Output the data in the kafka_sink stream to Kafka.

```
CREATE SINK STREAM kafka_sink (name STRING)
WITH (
  type="kafka",
  kafka_bootstrap_servers =  "ip1:port1,ip2:port2",
  kafka_topic = "testsink",
  encode = "json"
);
```

