ALIBABA CLOUD

Alibaba Cloud

Data Transmission Service

Change Tracking

Document Version: 20220617

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Prepare the source database account for change tracking

When you configure a change tracking task, you must specify the account of the source database. The database account is used for change tracking. Different databases require different permissions. You must create and authorize a database account before you configure a change tracking task.

Permissions required for the source database account

Database	Required permission	References
ApsaraDB RDS for MySQL instance	A database account that has read-only permissions, or a custom account that has the REPLICATION CLIENT, REPLICATION SLAVE, SHOW VIEW, and SELECT permissions	Create an account on an ApsaraDB RDS for MySQL instance and Modify the permissions of a standard account on an ApsaraDB RDS for MySQL instance
Self-managed MySQL database	The SELECT permission on the objects for change tracking The REPLICATION CLIENT, REPLICATION SLAVE, and SHOW VIEW permissions Permissions to create databases and tables. The permissions allow DTS to create a database named dts to record heartbeat data during change tracking.	Create an account for a user-created MySQL database and configure binary logging
ApsaraDB RDS for PostgreSQL instance	Permissions of a privileged account. The account must be the owner of the database.	Create an account on an ApsaraDB RDS for PostgreSQL instance and Create a database on an ApsaraDB RDS for PostgreSQL instance
PolarDB for MySQL cluster	A database account that has read-only permissions, or a custom account that has the REPLICATION CLIENT, REPLICATION SLAVE, SHOW VIEW, and SELECT permissions	Create a database account
PolarDB-X 1.0 instance	Read permissions on the objects for change tracking	Manage database accounts
Self-managed Oracle database	Permissions of the database administrator (DBA) **Notice** If the permissions of DBA cannot be granted to the database account, you can grant fine-grained permissions to the account. For more information, see Track data changes from a self-managed Oracle database.	CREATE USER and GRANT

2.Configure and manage change tracking tasks

2.1. Track data changes from an ApsaraDB RDS for MySQL instance

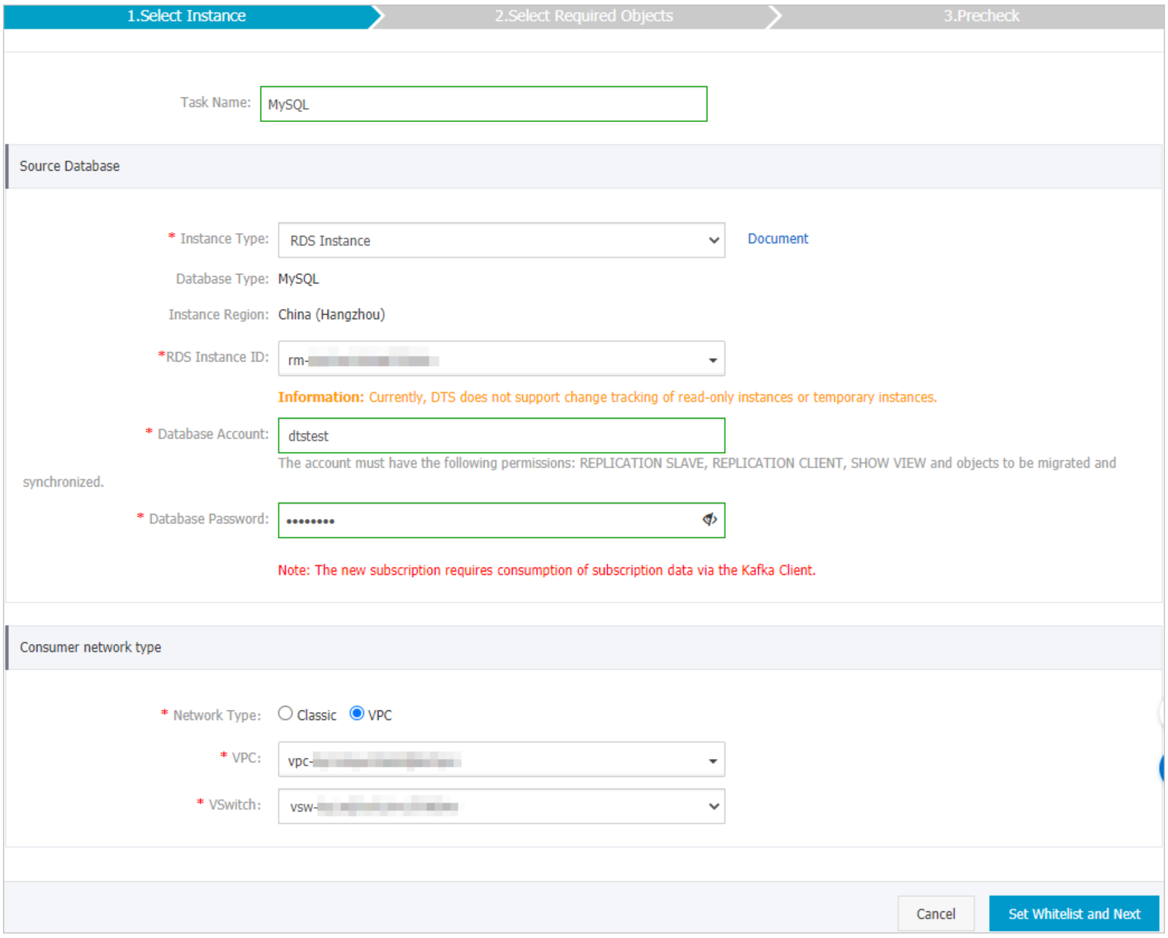
You can use Data Transmission Service (DTS) to track data changes from databases in real time. You can use the change tracking feature in the following scenarios: lightweight cache updates, business decoupling and asynchronous data processing, and synchronization of extract, transform, and load (ETL) operations. This topic describes how to track data changes from an ApsaraDB RDS for MySQL instance.

Precautions

- DTS does not track DDL operations that are performed by using gh-ost or pt-online-schema-change. Therefore, the change tracking client may fail to write the consumed data to the destination tables due to schema conflicts.
- If the source database is involved in another task such as a running data migration task, DTS may track data changes of other objects. In this case, you must manually filter the tracked data on the change tracking client.

Procedure

1. Create a change tracking instance. For more information, see Purchase a change tracking instance.
2. Log on to the DTS console.
3. In the left-side navigation pane, click **Change Tracking**.
4. In the upper part of the **Change Tracking Tasks** page, select the region where the change tracking instance resides.
5. Find the change tracking instance and click **Configure Task** in the Actions column.
6. Configure the source database and network type for the change tracking task.



Task Name: MySQL

Source Database

* Instance Type: RDS Instance Document

Database Type: MySQL

Instance Region: China (Hangzhou)

*RDS Instance ID: rm-[redacted]

Information: Currently, DTS does not support change tracking of read-only instances or temporary instances.

* Database Account: dtstest

The account must have the following permissions: REPLICATION SLAVE, REPLICATION CLIENT, SHOW VIEW and objects to be migrated and synchronized.

* Database Password: ••••••••

Note: The new subscription requires consumption of subscription data via the Kafka Client.

Consumer network type

* Network Type: ○ Classic ● VPC

* VPC: vpc-[redacted]

* VSwitch: vsw-[redacted]

Section	Parameter	Description
N/A	Task Name	The task name that DTS automatically generates. We recommend that you specify an descriptive name that makes it easy to identify the task. You do not need to use a unique task name.
	Instance Type	The type of the source instance. In this example, **RDS Instance** is selected for this parameter. **Note** If your source database is a self-managed database, you must deploy the network environment for the database. For more information, see Preparation overview.
	Database Type	The value of this parameter is set to **MySQL** and cannot be changed.
	Instance Region	The **source region** that you selected on the buy page. The value of this parameter cannot be changed.

Section	Parameter	Description
Source Database	RDS Instance ID	The ID of the ApsaraDB RDS for MySQL instance from which you want to track data changes. **Note** A read-only instance or temporary instance cannot be used as the source instance for change tracking.
	Database Account	The database account of the source ApsaraDB RDS for MySQL instance. **Note** ◦ The account must have the SELECT permission on the required objects and the REPLICATION CLIENT, REPLICATION SLAVE, and SHOW VIEW permissions. ◦ If the database engine of the source ApsaraDB RDS for MySQL instance is **MySQL 5.5** or **MySQL 5.6**, you do not need to configure the **database account** or **database password**.
	Database Password	The password for the account of the source ApsaraDB RDS for MySQL database.
Network type	◦ **Classic** ◦ **VPC**	The network type of the change tracking instance. **Note** ◦ If your change tracking client is deployed on a local server, you can select **Classic** or **VPC**. ◦ If your change tracking client is deployed on an ECS instance, we recommend that you select the network of the ECS instance. For example, if the ECS instance is deployed in a Virtual Private Cloud (VPC), select VPC as the network type and specify the **VPC** and **vSwitch** parameters. ◦ If you track data changes over internal networks, the network latency is minimal. ◦ **Classic** If you select **Classic**, no other configurations are required. For more information, see Network types. ◦ **VPC** If you select **VPC**, you must specify the **VPC** and **vSwitch** parameters. For more information, see Network types.

7. In the lower-right corner of the page, click **Set Whitelist and Next**.

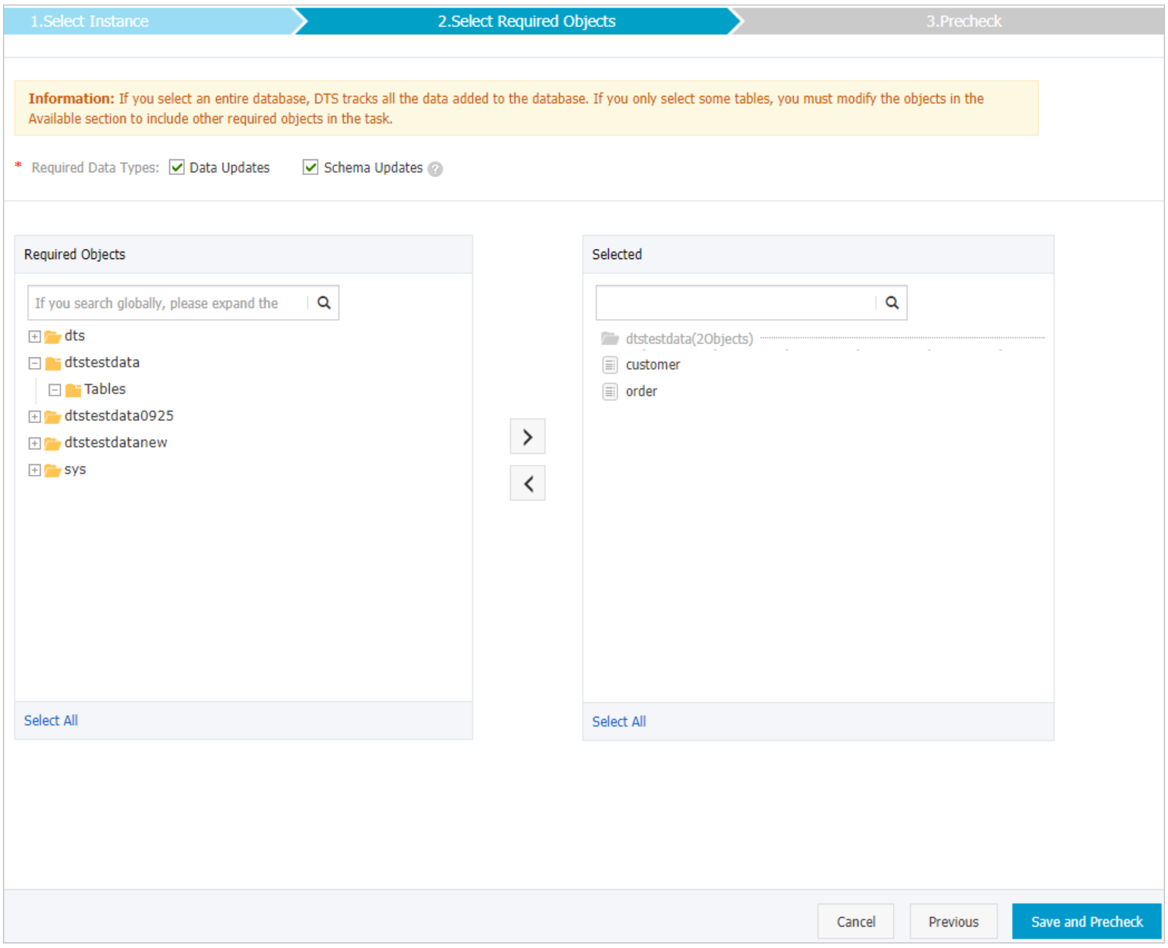
Warning

- If the source or destination database instance is an Alibaba Cloud database instance, such as an ApsaraDB RDS for MySQL or ApsaraDB for MongoDB instance, or is a self-managed database hosted on Elastic Compute Service (ECS), DTS automatically adds the CIDR blocks of DTS servers to the whitelist of the database instance or ECS security group rules. For more information, see Add the CIDR blocks of DTS servers to the security settings of on-premises databases. If the source or destination database is a self-managed database on data centers or is from other cloud service providers, you must manually add the CIDR blocks of DTS servers to allow DTS to access the database.
- If the CIDR blocks of DTS servers are automatically or manually added to the whitelist of the database instance or ECS security group rules, security risks may arise. Therefore, before you use DTS to migrate data, you must understand and acknowledge the potential risks and take preventive measures, including but not limited to the following measures: enhance the security of your account and password, limit the ports that are exposed, authenticate API calls, regularly check the whitelist or ECS security group rules and forbid unauthorized CIDR blocks, or connect the database to DTS by using Express Connect, VPN Gateway, or Smart Access Gateway.
- After your DTS task is completed or released, we recommend that you manually detect and remove the added CIDR blocks from the whitelist of the database instance or ECS security group rules.

8. In the **Create Change Tracking Account** message, wait until the account is created and then click **Next**.

   Note This step is required only if the database engine of the source ApsaraDB for MySQL instance is **MySQL 5.5** or **MySQL 5.6**. In this step, DTS creates a database account for change tracking in the source instance.

9. Select the data change types and objects.



Information: If you select an entire database, DTS tracks all the data added to the database. If you only select some tables, you must modify the objects in the Available section to include other required objects in the task.

* Required Data Types: ☑ Data Updates ☑ Schema Updates

Parameter	Description

Parameter	Description
Required Data Types	○ **Data Updates** DTS tracks data updates of the selected objects, including the INSERT, DELETE, and UPDATE operations. ○ **Schema Updates** DTS tracks the create, delete, and modify operations that are performed on all object schemas of the source instance. You must use the change tracking client to filter the required data. ? **Note** ○ If you select a database as the object, DTS tracks data changes of all objects, including new objects in the database. ○ If you select a table as the object, DTS tracks only data changes of this table. In this case, if you want to track data changes of another table, you must add the table to the selected objects. For more information, see Modify the objects for change tracking.
Required Objects	Select one or more objects from the **Required Objects** section and click the > icon to add the objects to the **Selected** section. ? **Note** You can select tables or databases as the objects for change tracking.

10. In the lower-right corner of the page, click **Save and Precheck**.

> **Note**
>
> - Before you can start the change tracking task, DTS performs a precheck. You can start the change tracking task only after the task passes the precheck.
> - If the task fails to pass the precheck, you can click the ⓘ icon next to each failed item to view details. After you troubleshoot the issues based on the causes, you can run a precheck again.

11. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed**.

After the change tracking task is configured, DTS performs initial change tracking, which takes about 1 minute. After initial change tracking is complete, you can create one or more consumer groups to consume the tracked data.

What to do next

1. Create consumer groups
2. Use a Kafka client to consume tracked data

2.2. Track data changes from a PolarDB for MySQL cluster

You can use Data Transmission Service (DTS) to track data changes in real time. This feature applies to the following scenarios: lightweight cache updates, business decoupling, asynchronous data processing, and synchronization of extract, transform, and load (ETL) operations. This topic describes how to track data changes from a PolarDB for MySQL cluster.

Prerequisites

The binary logging feature is enabled for the PolarDB for MySQL cluster. For more information, see Enable binary logging.

Precautions

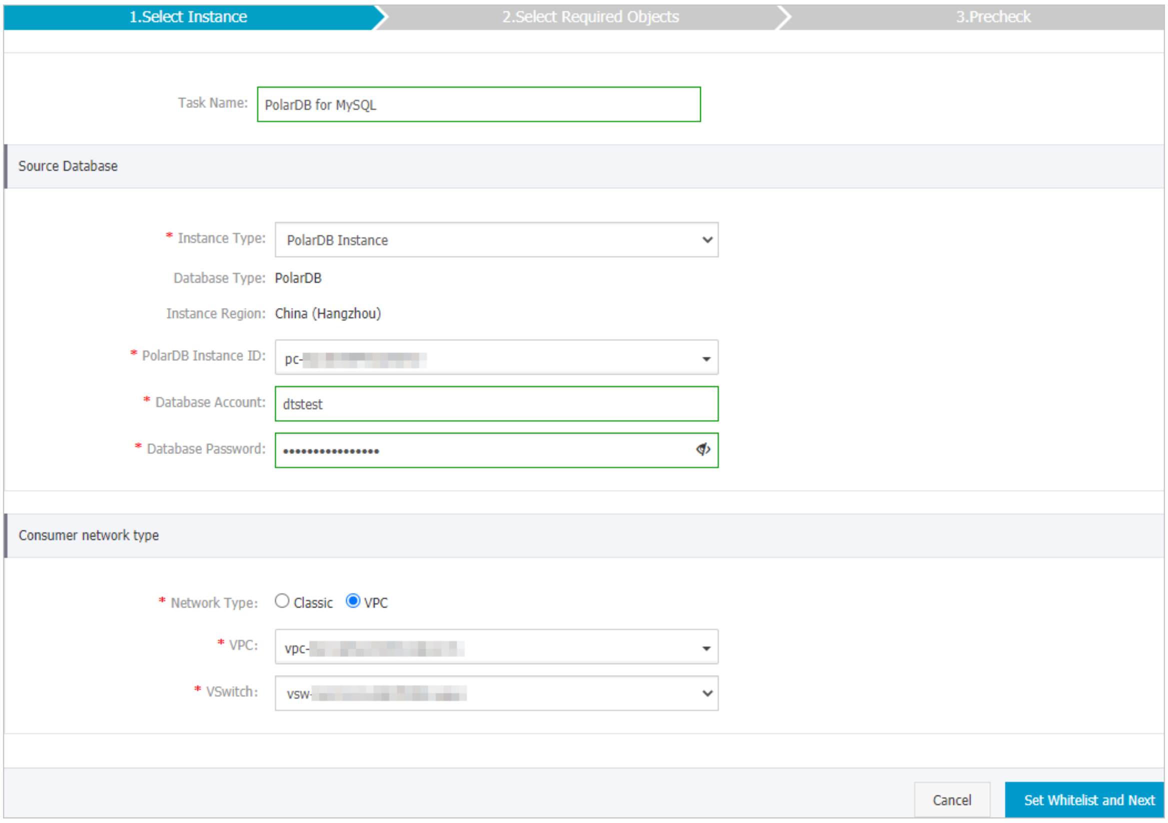
- DTS does not track DDL operations that are performed by using gh-ost or pt-online-schema-change. Therefore, the change tracking client may fail to write the consumed data to the destination tables due to schema conflicts.
- If the source database is involved in another task such as a running data migration task, DTS may track data changes of other objects. In this case, you must manually filter the tracked data on the change tracking client.

Procedure

1. Create a change tracking instance. For more information, see Purchase a change tracking instance.

   > ? **Note** On the buy page, set **Source Database** to **PolarDB** and select the region where the source PolarDB cluster resides.

2. Log on to the DTS console.
3. In the left-side navigation pane, click **Change Tracking**.
4. In the upper part of the **Change Tracking Tasks** page, select the region where the change tracking instance resides.
5. Find the change tracking instance, and click **Configure Channel** in the Actions column.
6. Configure the source database and network type for the change tracking task.



Task Name: PolarDB for MySQL

Source Database

* Instance Type: PolarDB Instance

Database Type: PolarDB

Instance Region: China (Hangzhou)

* PolarDB Instance ID: pc-[redacted]

* Database Account: dtstest

* Database Password: ••••••••••••••••

Consumer network type

* Network Type: ○ Classic ● VPC

* VPC: vpc-[redacted]

* VSwitch: vsw-[redacted]

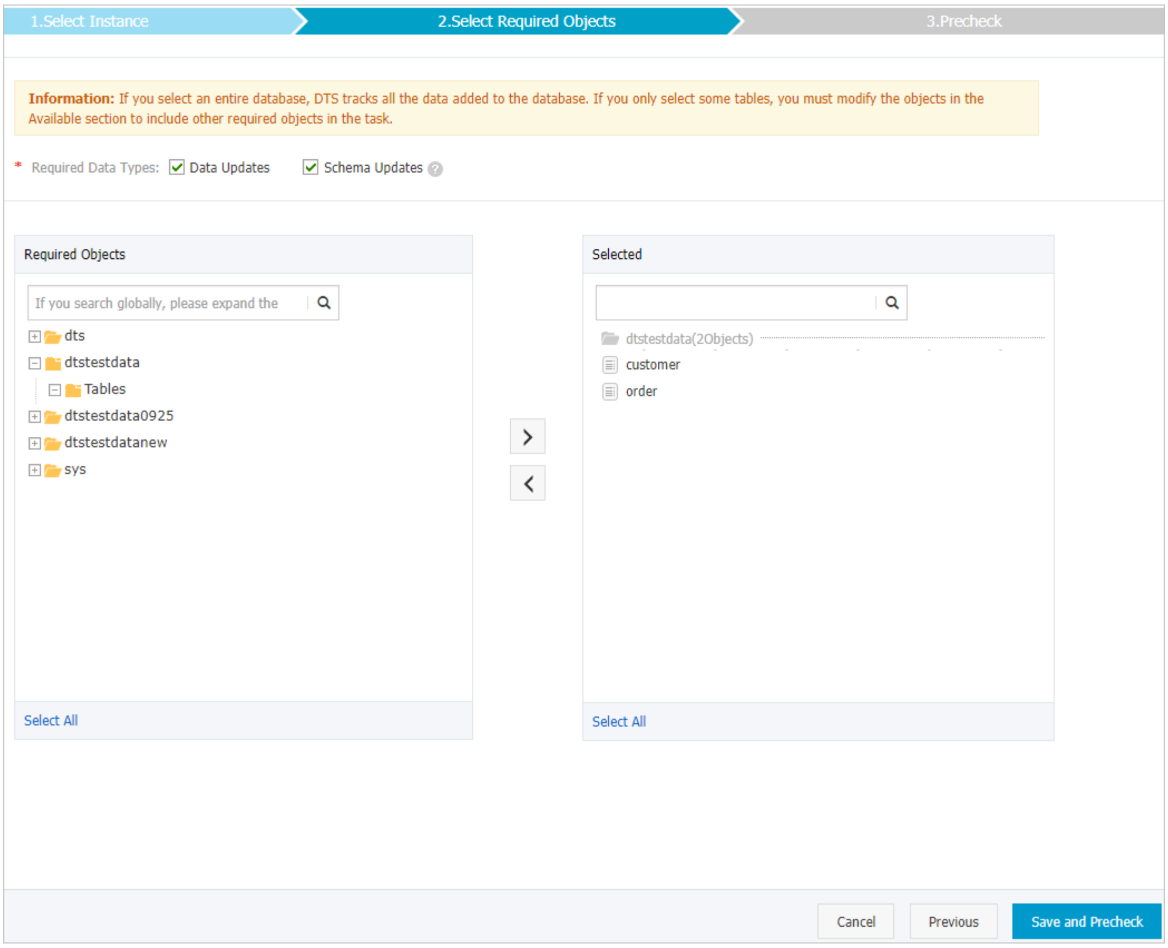
Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	The value of this parameter is set to **PolarDB Instance** and cannot be changed.
	Database Type	The value of this parameter is set to **PolarDB** and cannot be changed.
	Instance Region	The **source region** that you selected on the buy page. You cannot change the value of this parameter.
	PolarDB Instance ID	Select the ID of the source PolarDB cluster.
	Database Account	Enter the database account of the source PolarDB cluster. **Note** The account must have the SELECT permission on the required objects, the REPLICATION CLIENT permission, the REPLICATION SLAVE permission, and the SHOW VIEW permission.
	Database Password	Enter the password of the database account.

Section	Parameter	Description
Network Type	N/A	Select the network type of the change tracking instance. ○ **Classic** If you select **Classic**, no other configurations are required. For more information, see Classic network. ○ **VPC** If you select **VPC**, you must specify the **VPC** and **vSwitch** parameters. For more information, see VPC. **Note** ○ We recommend that you select the same network type as the ECS instance on which the change tracking client is installed. For example, if the ECS instance is deployed in a virtual private cloud (VPC), select VPC as the network type and specify the **VPC** and **vSwitch** parameters. ○ If you track data changes over internal networks, the network latency is minimal.

7. In the lower-right corner of the page, click **Set Whitelist and Next**.

   > **Note** DTS adds the CIDR blocks of DTS servers to the whitelist of the source PolarDB cluster. This ensures that DTS servers can connect to the source PolarDB cluster.

8. Select the data change types and objects.



1.Select Instance

2.Select Required Objects

3.Precheck

Information: If you select an entire database, DTS tracks all the data added to the database. If you only select some tables, you must modify the objects in the Available section to include other required objects in the task.

* Required Data Types: ☑ Data Updates ☑ Schema Updates

Required Objects

If you search globally, please expand the

- dts
- dtstestdata
  - Tables
- dtstestdata0925
- dtstestdatanew
- sys

Selected

- dtstestdata(2Objects)
  - customer
  - order

Parameter	Description

Parameter	Description
Required Data Types	○ **Data Updates** DTS tracks data updates of the selected objects, including the INSERT, DELETE, and UPDATE operations. ○ **Schema Updates** DTS tracks the create, delete, and modify operations that are performed on all object schemas of the source instance. You must use the change tracking client to filter the required data. **Note** ○ If you select a database as the object, DTS tracks data changes of all objects, including new objects in the database. ○ If you select a table as the object, DTS tracks only data changes of this table. In this case, if you want to track data changes of another table, you must add the table to the selected objects. For more information, see Modify the objects for change tracking.
Required Objects	Select one or more objects from the **Required Objects** section and click the › icon to add the objects to the **Selected** section. **Note** You can select tables or databases as the objects for change tracking.

9. In the lower-right corner of the page, click **Save and Precheck**.

> **Note**
>
> - Before you can start the change tracking task, DTS performs a precheck. You can start the change tracking task only after the task passes the precheck.
> - If the task fails to pass the precheck, you can click the ⓘ icon next to each failed item to view details. After you troubleshoot the issues based on the causes, you can run a precheck again.

10. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed**.

What to do next

After the change tracking task is configured, DTS performs initial change tracking, which takes about 1 minute. After initial change tracking is completed, you can perform the following operations:

1. Create consumer groups
2. Use a Kafka client to consume tracked data
3. Use the SDK demo code to consume tracked data

2.3. Track data changes from a self-managed Oracle database

Data Transmission Service (DTS) allows you to track data changes from databases in real time. You can use the change tracking feature in the following scenarios: lightweight cache updates, business decoupling and asynchronous data processing, and synchronization of extract, transform, and load (ETL) operations. This topic describes how to track data changes from a self-managed Oracle database.

Prerequisites

- The engine version of the self-managed Oracle database is 9i, 10g, or 11g. Real Application Cluster (RAC) instances are not supported.
- Supplemental logging, including SUPPLEMENTAL_LOG_DATA_PK and SUPPLEMENTAL_LOG_DATA_UI, is enabled for the self-managed Oracle database. For more information, see Supplemental Logging.
- The self-managed Oracle database is running in ARCHIVELOG mode. Archived log files are accessible and a suitable retention period is set for archived log files. For more information, see Managing Archived Redo Log Files.

Precautions

- DTS does not track DDL operations that are performed by using gh-ost or pt-online-schema-change. Therefore, the change tracking client may fail to write the consumed data to the destination tables due to schema conflicts.
- If the source database is involved in another task such as a running data migration task, DTS may track data changes of other objects. In this case, you must manually filter the tracked data on the change tracking client.

Preparations

Log on to the source Oracle database, create a database account, and grant the database administrator (DBA) permission to the account.

Notice

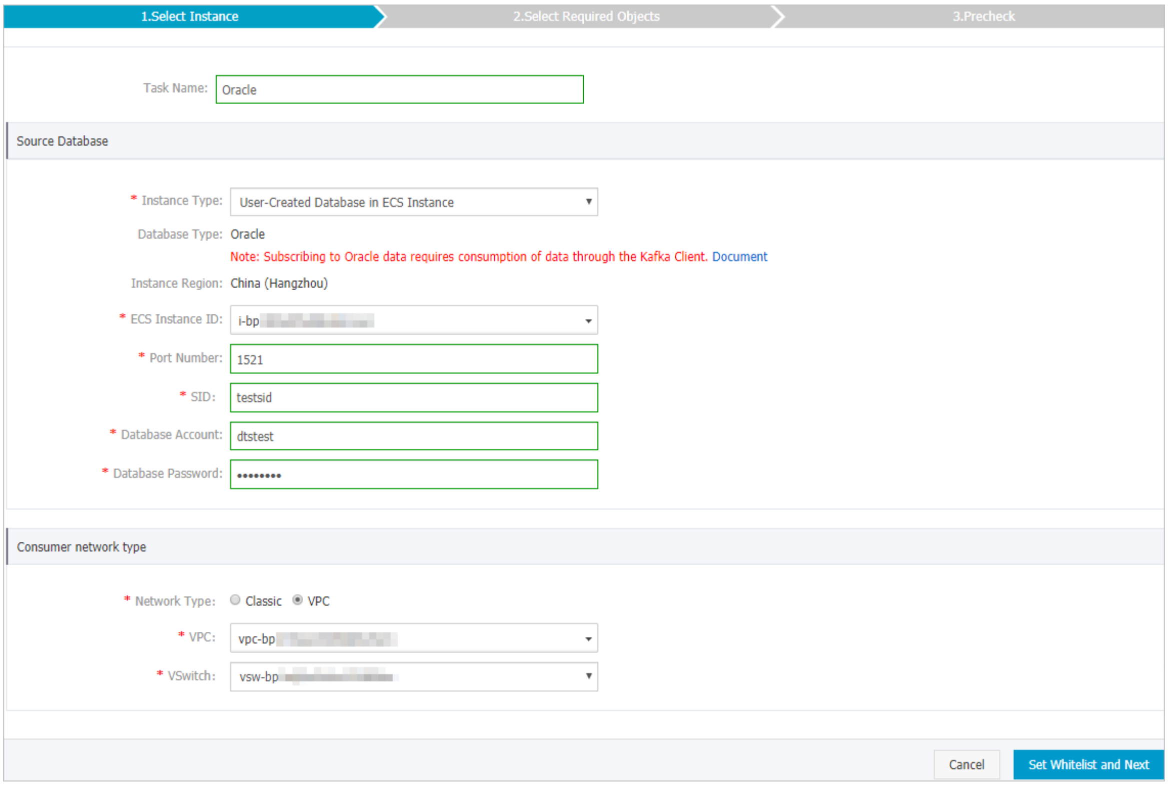
Procedure

1. Create a change tracking instance. For more information, see Purchase procedure.

   Note On the buy page, set **Instance Type** to **Oracle** and select the region where you want to create the instance.

2. Log on to the DTS console.

3. In the left-side navigation pane, click **Change Tracking**.
4. In the upper part of the **Change Tracking Tasks** page, select the region where the change tracking instance resides.
5. Find the change tracking instance and click **Configure Task** in the Actions column.
6. Configure the source database and network type for the change tracking task.



Task Name: Oracle

Source Database

* Instance Type: User-Created Database in ECS Instance

Database Type: Oracle

Note: Subscribing to Oracle data requires consumption of data through the Kafka Client. Document

Instance Region: China (Hangzhou)

* ECS Instance ID: i-bp[redacted]

* Port Number: 1521

* SID: testsid

* Database Account: dtstest

* Database Password: ••••••••

Consumer network type

* Network Type: ○ Classic ● VPC

* VPC: vpc-bp[redacted]

* VSwitch: vsw-bp[redacted]

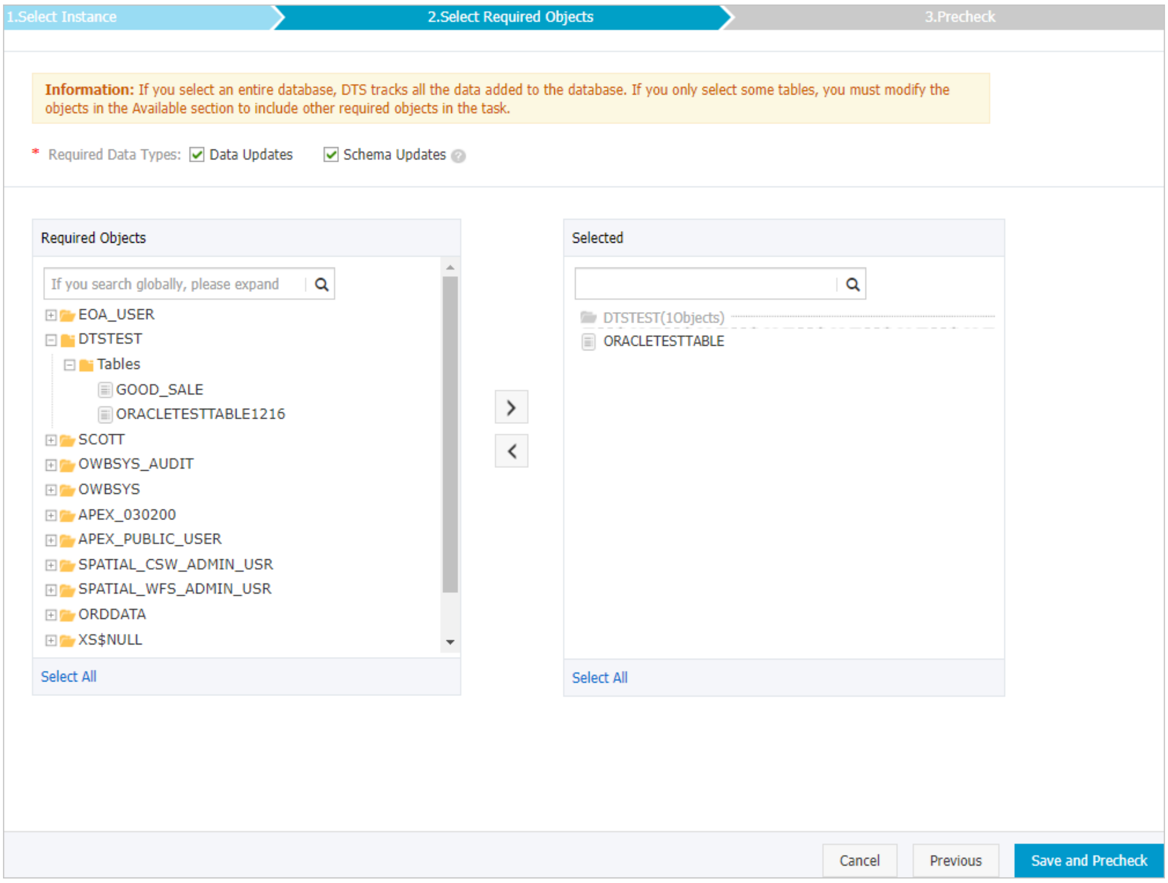
Section	Parameter	Description
None	Task Name	The task name that DTS automatically generates. We recommend that you specify a descriptive name that makes it easy to identify the task. You do not need to use a unique task name.
Source Database	Instance type	The type of the source instance. In this example, **User-Created Database in ECS Instance** is used. **Note** If you select other instance types, you must deploy the network environment for the self-managed database. For more information, see Preparation overview.
	Database Type	The value of this parameter is set to Oracle and cannot be changed.
	Instance Region	The **source region** that you selected on the buy page. The value of this parameter cannot be changed.
	ECS Instance ID	The ID of the Elastic Compute Service (ECS) instance that hosts the self-managed Oracle database.

Section	Parameter	Description
	Port Number	The service port number of the self-managed Oracle database.
	SID	The system ID (SID) of the self-managed Oracle database.
	Database Account	The account of the self-managed Oracle database. **Note** For more information about how to create and authorize a database account, see Preparations.
	Database Password	The password of the database account.
Consumer Network Type	None	The network type of the change tracking instance. **Note** ◦ We recommend that you select the same network type as the ECS instance on which the change tracking client is installed. For example, if the ECS instance is deployed in a virtual private cloud (VPC), select VPC as the network type and specify the **VPC** and **vSwitch** parameters. ◦ If you track data changes over internal networks, the network latency is minimal. ◦ **Classic** If you select **Classic**, no other configurations are required. For more information, see Classic network. ◦ **VPC** If you select **VPC**, you must specify the **VPC** and **vSwitch** parameters. For more information, see VPC.

7. In the lower-right corner of the page, click **Set Whitelist and Next**.

> **Warning**
>
> - If the source or destination database instance is an Alibaba Cloud database instance, such as an ApsaraDB RDS for MySQL or ApsaraDB for MongoDB instance, or is a self-managed database hosted on Elastic Compute Service (ECS), DTS automatically adds the CIDR blocks of DTS servers to the whitelist of the database instance or ECS security group rules. For more information, see Add the CIDR blocks of DTS servers to the security settings of on-premises databases. If the source or destination database is a self-managed database on data centers or is from other cloud service providers, you must manually add the CIDR blocks of DTS servers to allow DTS to access the database.
> - If the CIDR blocks of DTS servers are automatically or manually added to the whitelist of the database instance or ECS security group rules, security risks may arise. Therefore, before you use DTS to migrate data, you must understand and acknowledge the potential risks and take preventive measures, including but not limited to the following measures: enhance the security of your account and password, limit the ports that are exposed, authenticate API calls, regularly check the whitelist or ECS security group rules and forbid unauthorized CIDR blocks, or connect the database to DTS by using Express Connect, VPN Gateway, or Smart Access Gateway.
> - After your DTS task is completed or released, we recommend that you manually detect and remove the added CIDR blocks from the whitelist of the database instance or ECS security group rules.

8. Select the data change types and objects.



Information: If you select an entire database, DTS tracks all the data added to the database. If you only select some tables, you must modify the objects in the Available section to include other required objects in the task.

* Required Data Types: ☑ Data Updates ☑ Schema Updates

Parameter	Description
Required Data Types	○ **Data Updates** DTS tracks data updates of the selected objects, including the INSERT, DELETE, and UPDATE operations. ○ **Schema Updates** DTS tracks the create, delete, and modify operations that are performed on all object schemas of the source instance. You must use the change tracking client to filter the required data. **Note** ○ If you select a database as the object, DTS tracks data changes of all objects, including new objects in the database. ○ If you select a table as the object, DTS tracks only data changes of this table. In this case, if you want to track data changes of another table, you must add the table to the selected objects. For more information, see Modify the objects for change tracking.
Required Objects	Select one or more objects from the **Required Objects** section and click the > icon to add the objects to the **Selected** section. **Note** You can select tables or databases as the objects for change tracking.

9. In the lower-right corner of the page, click **Save and Precheck**.

   Note

   - Before you can start the change tracking task, DTS performs a precheck. You can start the change tracking task only after the task passes the precheck.
   - If the task fails to pass the precheck, you can click the ⓘ icon next to each failed item to view details. After you troubleshoot the issues based on the causes, you can run a precheck again.

10. Close the **Precheck** dialog box after the following message is displayed: **Precheck Passed**. After the change tracking task is configured, DTS performs initial change tracking. Initial change tracking takes about one minute. After initial change tracking is complete, you can create one or more consumer groups to consume the tracked data.

What's next

1. Create consumer groups
2. Use a Kafka client to consume tracked data
3. Use the SDK demo code to consume tracked data

2.4. Modify the objects for change tracking

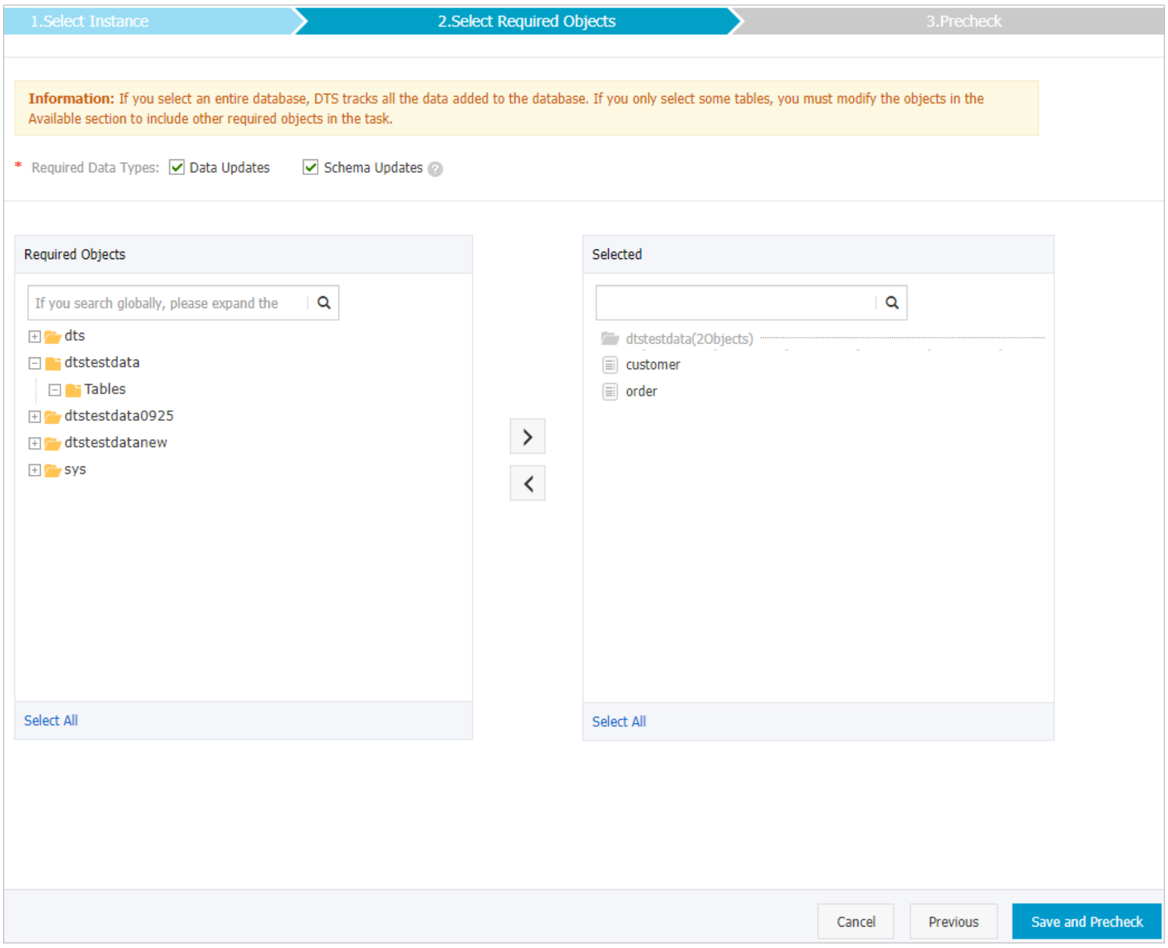
Data Transmission Service (DTS) allows you to add or remove the objects for change tracking in the consumption process. This topic describes how to modify the objects for change tracking.

Precautions

- After you add an object, the change tracking task pulls the data changes of the new object from the time when the modification takes effect.
- If the change tracking client tracks the data changes of a removed object, you must filter the tracked data changes on the client.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. In the upper part of the **Change Tracking Tasks** page, select the region where the change tracking instance resides.
4. Find the change tracking instance and click **Modify Required Objects** in the **Actions** column.
5. In the **Select Required Objects** step, you can add or remove the objects for change tracking.



Information: If you select an entire database, DTS tracks all the data added to the database. If you only select some tables, you must modify the objects in the Available section to include other required objects in the task.

* Required Data Types: ☑ Data Updates ☑ Schema Updates

- Add the objects for change tracking

  In the **Required Objects** section, select one or more objects and click the > icon to add the objects to the **Selected** section.

- Remove the objects for change tracking

  In the **Selected** section, select one or more objects and click the < icon to move the objects to the **Required Objects** section.

6. In the lower-right corner of the page, click **Save and Precheck**.

   > ? Note
   >
   > - Before you can start the change tracking task, a precheck is performed. You can start the change tracking task only after the task passes the precheck.
   > - If the task fails to pass the precheck, click the ⓘ icon next to each failed item to view details. Troubleshoot the issues based on the causes and run a precheck again.

7. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**

After the change tracking task is configured, DTS performs initial change tracking, which takes about 1 minute. After the initial change tracking is complete, you can use the tracked data changes for subsequent operations. For more information, see Use a Kafka client to consume tracked data.

2.5. View tracked data changes

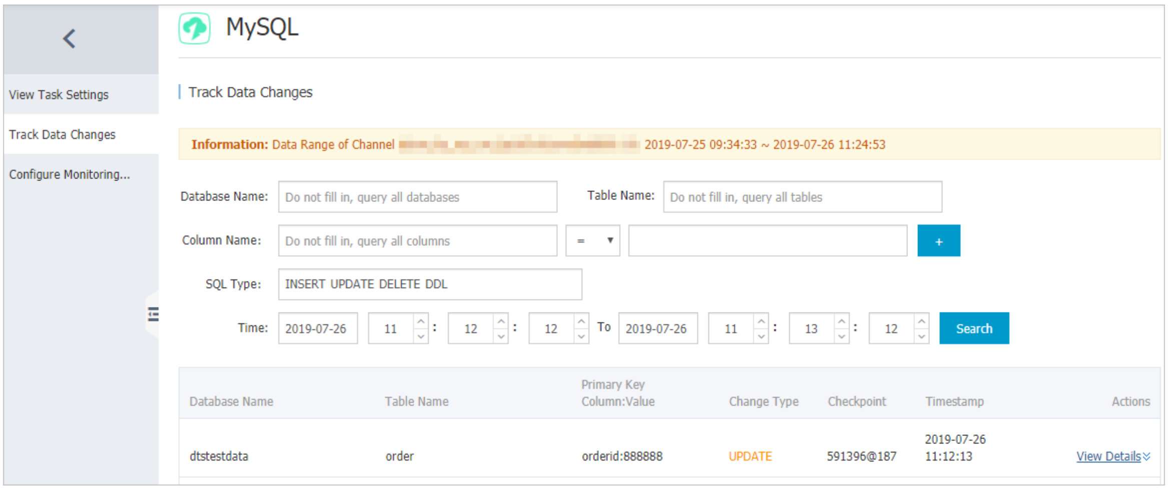
This topic describes how to view the incremental data from a change tracking task in the DTS console. This topic provides the definition of each field in the tracked data changes.

Prerequisites

- A change tracking task is created. For more information, see Track data changes from an ApsaraDB RDS for MySQL instance.
- The change tracking task is in the Normal or Error state.

Procedure

1.
2.
3. At the top of the **Change Tracking Tasks** page, select the region where the change tracking instance resides.
4. Find the change tracking instance and click the instance ID.
5. In the left-side navigation pane, click **Track Data Changes**.
6. On the **Track Data Changes** page, 20 data entries that are tracked in the last minute are displayed by default. You can specify conditions to filter the tracked data changes.



View Task Settings
Track Data Changes
Configure Monitoring...

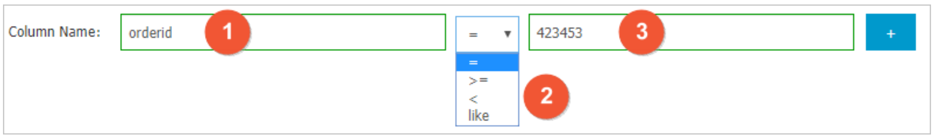
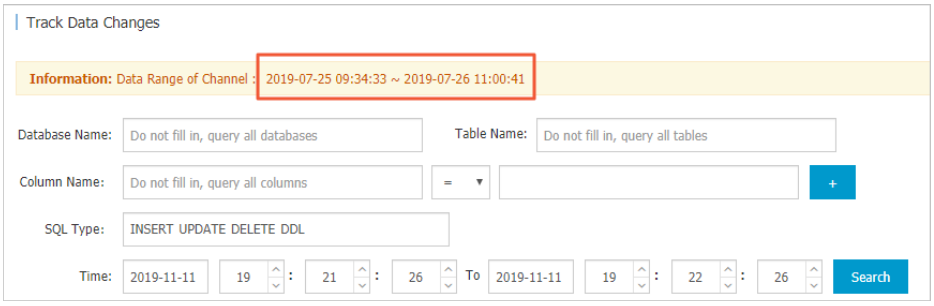
MySQL

Track Data Changes

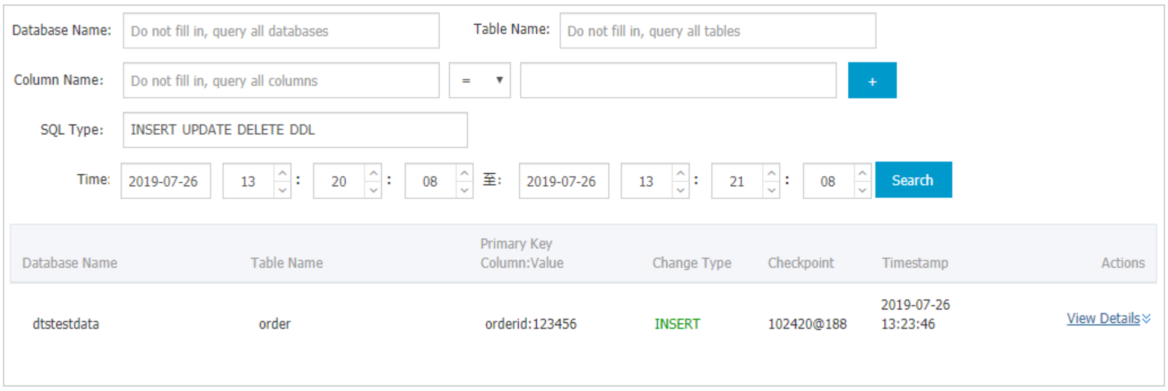
Information: Data Range of Channel [redacted] 2019-07-25 09:34:33 ~ 2019-07-26 11:24:53

Database Name: Do not fill in, query all databases
Table Name: Do not fill in, query all tables
Column Name: Do not fill in, query all columns =
SQL Type: INSERT UPDATE DELETE DDL
Time: 2019-07-26 11 : 12 : 12 To 2019-07-26 11 : 13 : 12 Search

Database Name	Table Name	Primary Key Column:Value	Change Type	Checkpoint	Timestamp	Actions
dtstestdata	order	orderid:888888	UPDATE	591396@187	2019-07-26 11:12:13	View Details

Filter condition	Description
Database Name	Enter the name of the database that you want to query. If you do not specify this condition, all databases are queried.
Table Name	Enter the name of the table that you want to query. If you do not specify this condition, all tables are queried.
Column Name	Enter the name of the column that you want to query, select an operator, and then enter the column value. If you do not specify this condition, all columns are queried. **Note** You can click the plus sign (+) to add more column filters.  Column Name: orderid 1 = 423453 3 + = >= < like 2
SQL Type	Select the types of SQL operations that you want to query. INSERT, UPDATE, DELETE, and DDL operations are queried by default.
Time	Select a time range. The selected time range must be within the time range of all tracked data changes. For more information, see the prompt at the top of the page.  Track Data Changes Information: Data Range of Channel : 2019-07-25 09:34:33 ~ 2019-07-26 11:00:41 Database Name: Do not fill in, query all databases Table Name: Do not fill in, query all tables Column Name: Do not fill in, query all columns = + SQL Type: INSERT UPDATE DELETE DDL Time: 2019-11-11 19 : 21 : 26 To 2019-11-11 19 : 22 : 26 Search

7. Click **Search** to retrieve specific data entries.



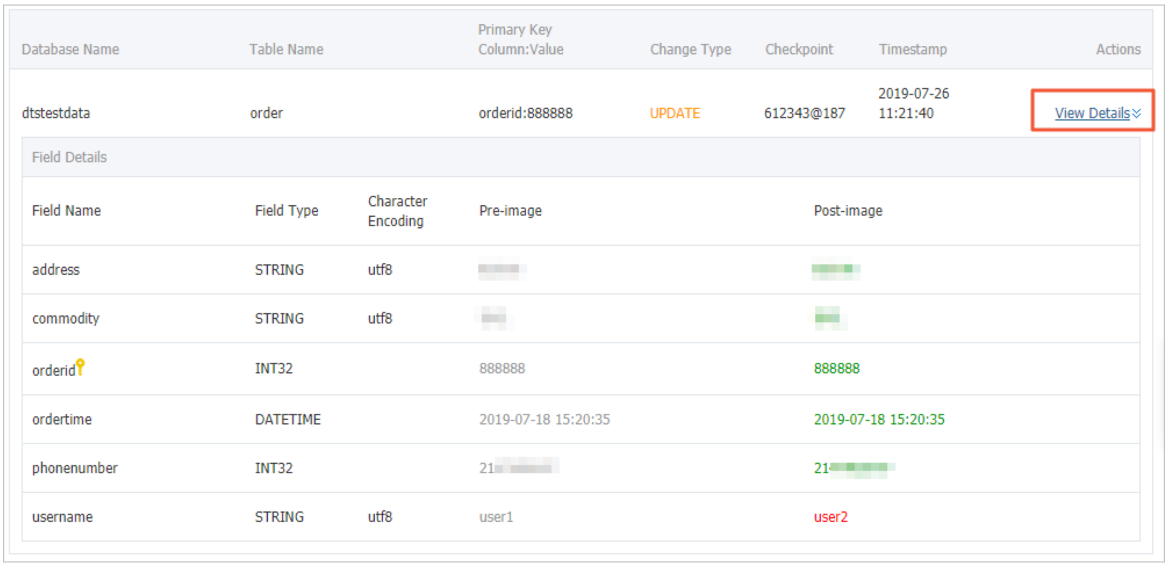
Database Name: Do not fill in, query all databases
Table Name: Do not fill in, query all tables
Column Name: Do not fill in, query all columns = ▾ +
SQL Type: INSERT UPDATE DELETE DDL
Time: 2019-07-26 13 : 20 : 08 至: 2019-07-26 13 : 21 : 08 Search

Database Name	Table Name	Primary Key Column:Value	Change Type	Checkpoint	Timestamp	Actions
dtstestdata	order	orderid:123456	INSERT	102420@188	2019-07-26 13:23:46	View Details

Consumption checkpoint and timestamp

Database type	Consumption checkpoint	Timestamp
ApsaraDB RDS for MySQL	The location and time of an incremental data entry in the binary log.	The timestamp when an incremental data entry is written to the binary log file.
Oracle	The location and time of an incremental data entry in the redo or archive log.	The timestamp when an incremental data entry is written to the redo or archive log.

8. Click **View Details** in the Actions column of a data entry to view the details.



Database Name	Table Name	Primary Key Column:Value	Change Type	Checkpoint	Timestamp	Actions
dtstestdata	order	orderid:888888	UPDATE	612343@187	2019-07-26 11:21:40	View Details

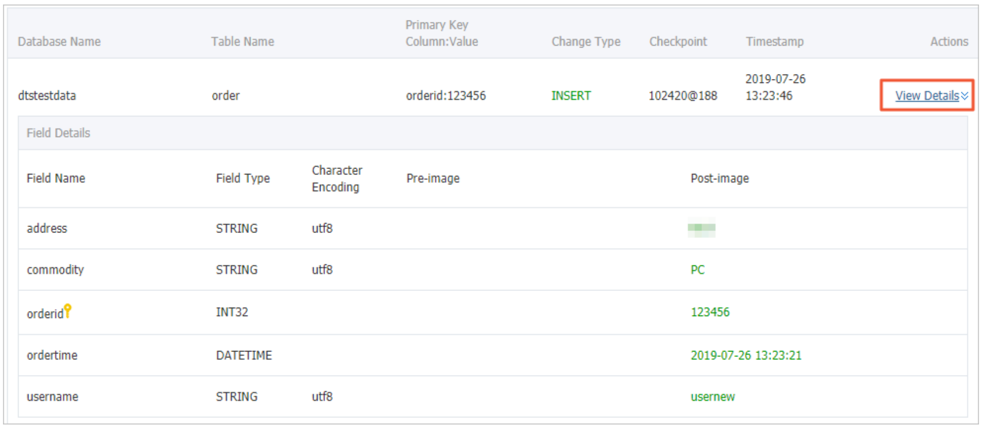
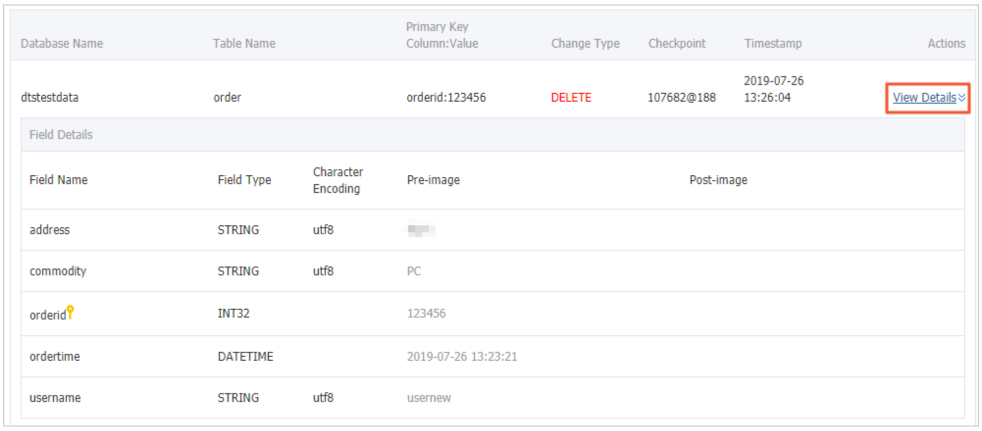
Field Details

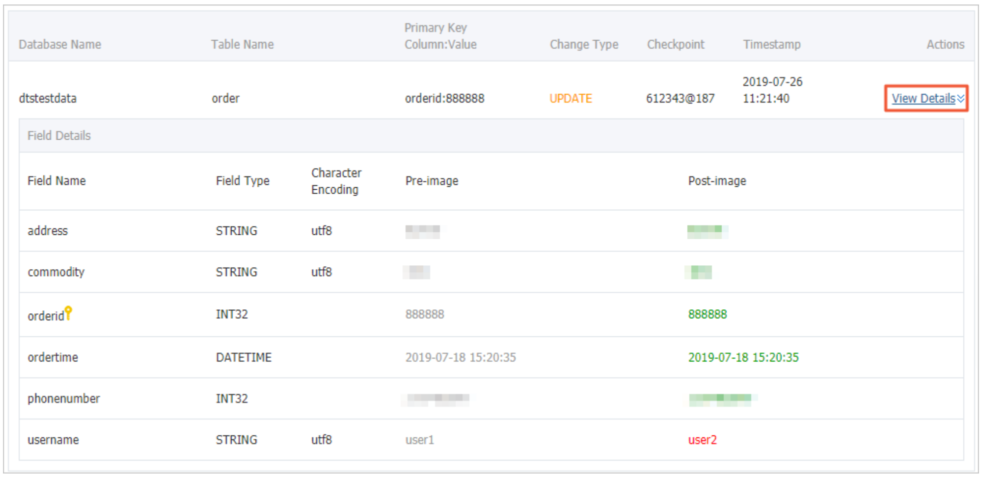
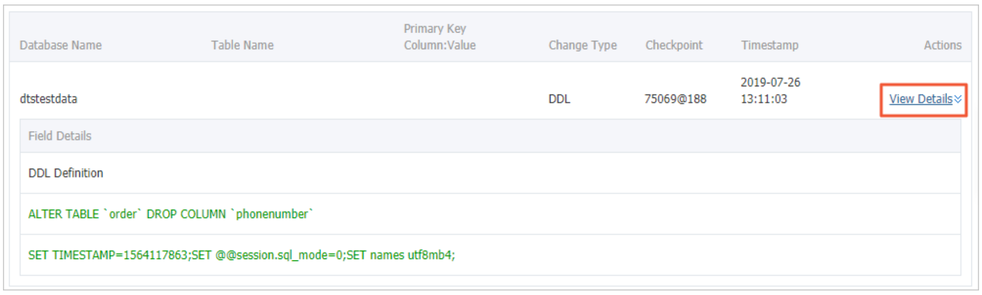
Field Name	Field Type	Character Encoding	Pre-image	Post-image
address	STRING	utf8	[redacted]	[redacted]
commodity	STRING	utf8	[redacted]	[redacted]
orderid	INT32		888888	888888
ordertime	DATETIME		2019-07-18 15:20:35	2019-07-18 15:20:35
phonenumber	INT32		21[redacted]	21[redacted]
username	STRING	utf8	user1	user2

Field	Description
Field Name	The name of the field or column.
Field Type	The type of the field.
Character Encoding	The character encoding of the field, such as UTF-8, GBK, Latin 1, or UTF-8 (MB4).

Field	Description
Pre-image	The value of each field before the data entry is updated.
Post-image	The value of each field after the data entry is updated. The updated values are displayed in red.

Additional information about the pre-image and post-image

<table>
<tr><th>Operation type</th><th>Value description</th></tr>
<tr><td>INSERT</td><td>

<table>
<tr><th>Database Name</th><th>Table Name</th><th>Primary Key Column:Value</th><th>Change Type</th><th>Checkpoint</th><th>Timestamp</th><th>Actions</th></tr>
<tr><td>dtstestdata</td><td>order</td><td>orderid:123456</td><td>INSERT</td><td>102420@188</td><td>2019-07-26 13:23:46</td><td>View Details</td></tr>
</table>
Field Details
<table>
<tr><th>Field Name</th><th>Field Type</th><th>Character Encoding</th><th>Pre-image</th><th>Post-image</th></tr>
<tr><td>address</td><td>STRING</td><td>utf8</td><td></td><td>[redacted]</td></tr>
<tr><td>commodity</td><td>STRING</td><td>utf8</td><td></td><td>PC</td></tr>
<tr><td>orderid</td><td>INT32</td><td></td><td></td><td>123456</td></tr>
<tr><td>ordertime</td><td>DATETIME</td><td></td><td></td><td>2019-07-26 13:23:21</td></tr>
<tr><td>username</td><td>STRING</td><td>utf8</td><td></td><td>usernew</td></tr>
</table>
The value of the pre-image is empty.
The value of the post-image is the data written by the INSERT operation.
</td></tr>
<tr><td>DELETE</td><td>

<table>
<tr><th>Database Name</th><th>Table Name</th><th>Primary Key Column:Value</th><th>Change Type</th><th>Checkpoint</th><th>Timestamp</th><th>Actions</th></tr>
<tr><td>dtstestdata</td><td>order</td><td>orderid:123456</td><td>DELETE</td><td>107682@188</td><td>2019-07-26 13:26:04</td><td>View Details</td></tr>
</table>
Field Details
<table>
<tr><th>Field Name</th><th>Field Type</th><th>Character Encoding</th><th>Pre-image</th><th>Post-image</th></tr>
<tr><td>address</td><td>STRING</td><td>utf8</td><td>[redacted]</td><td></td></tr>
<tr><td>commodity</td><td>STRING</td><td>utf8</td><td>PC</td><td></td></tr>
<tr><td>orderid</td><td>INT32</td><td></td><td>123456</td><td></td></tr>
<tr><td>ordertime</td><td>DATETIME</td><td></td><td>2019-07-26 13:23:21</td><td></td></tr>
<tr><td>username</td><td>STRING</td><td>utf8</td><td>usernew</td><td></td></tr>
</table>
The value of the pre-image is the data deleted by the DELETE operation.
The value of the post-image is empty.
</td></tr>
</table>

<table>
<tr><th>Operation type</th><th>Value description</th></tr>
<tr><td>UPDATE</td><td>

<table>
<tr><th>Database Name</th><th>Table Name</th><th>Primary Key Column:Value</th><th>Change Type</th><th>Checkpoint</th><th>Timestamp</th><th>Actions</th></tr>
<tr><td>dtstestdata</td><td>order</td><td>orderid:888888</td><td>UPDATE</td><td>612343@187</td><td>2019-07-26 11:21:40</td><td>View Details</td></tr>
</table>
Field Details
<table>
<tr><th>Field Name</th><th>Field Type</th><th>Character Encoding</th><th>Pre-image</th><th>Post-image</th></tr>
<tr><td>address</td><td>STRING</td><td>utf8</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>commodity</td><td>STRING</td><td>utf8</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>orderid</td><td>INT32</td><td></td><td>888888</td><td>888888</td></tr>
<tr><td>ordertime</td><td>DATETIME</td><td></td><td>2019-07-18 15:20:35</td><td>2019-07-18 15:20:35</td></tr>
<tr><td>phonenumber</td><td>INT32</td><td></td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>username</td><td>STRING</td><td>utf8</td><td>user1</td><td>user2</td></tr>
</table>
The value of the pre-image is the data before update.
The value of the post-image is the data after update.
Note The updated values are displayed in red.
</td></tr>
<tr><td>DDL or DML</td><td>

<table>
<tr><th>Database Name</th><th>Table Name</th><th>Primary Key Column:Value</th><th>Change Type</th><th>Checkpoint</th><th>Timestamp</th><th>Actions</th></tr>
<tr><td>dtstestdata</td><td></td><td></td><td>DDL</td><td>75069@188</td><td>2019-07-26 13:11:03</td><td>View Details</td></tr>
</table>
Field Details
DDL Definition
ALTER TABLE `order` DROP COLUMN `phonenumber`
SET TIMESTAMP=1564117863;SET @@session.sql_mode=0;SET names utf8mb4;
The details only contain DDL or DML statements.
</td></tr>
</table>

2.6. Reset a change tracking task

This topic describes how to reset a change tracking task. You can reset a change tracking task to clear the configurations of the task and delete the data that is cached by the task.

Prerequisites

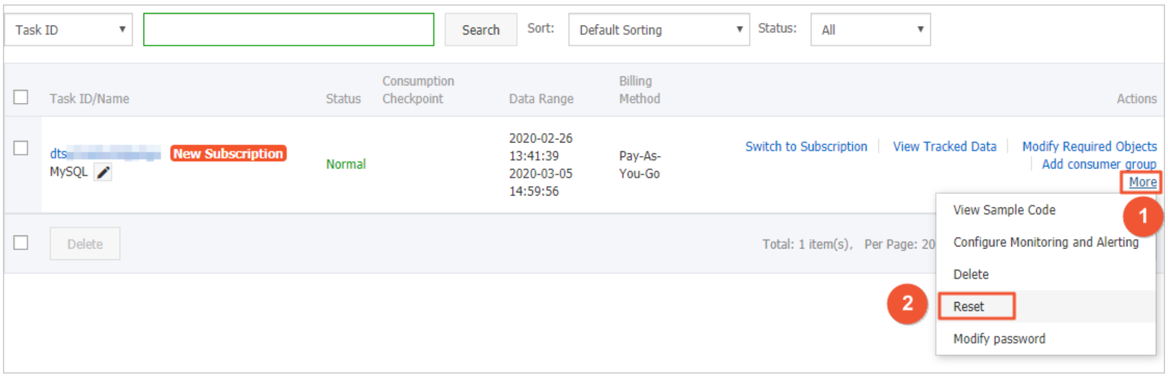
The change tracking task is in the **Normal**, **Initial synchronizing**, or **Error** state.

Impacts on billing

- Subscription: no impact.
- Pay-as-you-go: The change tracking task will enter the **Not Configured** state. You are not billed for the task when it is in this state. The billing restarts only after you configure and start the change tracking task.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. At the top of the **Change Tracking** page, select the region where your change tracking instance resides.
4. Find the target change tracking task, and choose **More > Reset** in the **Actions** column.



☐	Task ID/Name	Status	Consumption Checkpoint	Data Range	Billing Method	Actions
☐	dts[redacted] New Subscription MySQL	Normal		2020-02-26 13:41:39 2020-03-05 14:59:56	Pay-As-You-Go	Switch to Subscription | View Tracked Data | Modify Required Objects | Add consumer group More

Total: 1 item(s), Per Page: 20

> **Warning** Resetting a change tracking task has the following impacts. Proceed with caution.
>
> - The change tracking task stops tracking data changes from the source database.
> - The configurations of the change tracking task are deleted. The status of the change tracking task changes to **Not Configured**.
> - Data changes that are cached by the change tracking task are deleted and cannot be restored.

5. In the dialog box that appears, click **OK**.

References

For information about how to configure a change tracking task, see Overview of change tracking scenarios.

3.Consume tracked data

3.1. Create consumer groups

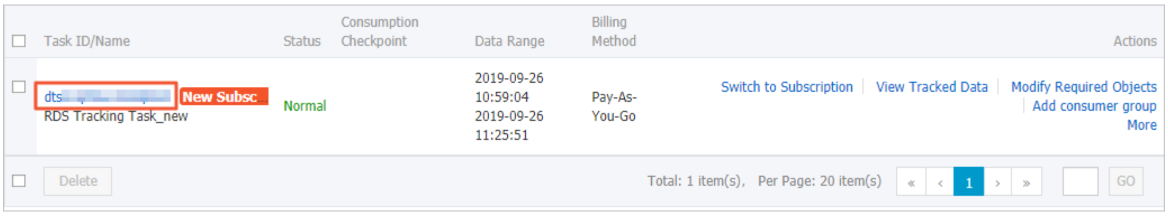
You can create multiple consumer groups when you use the change tracking feature. Consumers in different consumer groups can track data changes from the same data source. Consumer groups allow you to reduce usage costs and improve the efficiency of data consumption.

Precautions

- You can create a maximum of 20 consumer groups in a change tracking instance to achieve repeated data consumption.
- You can create only one consumer in each consumer group.

Procedure

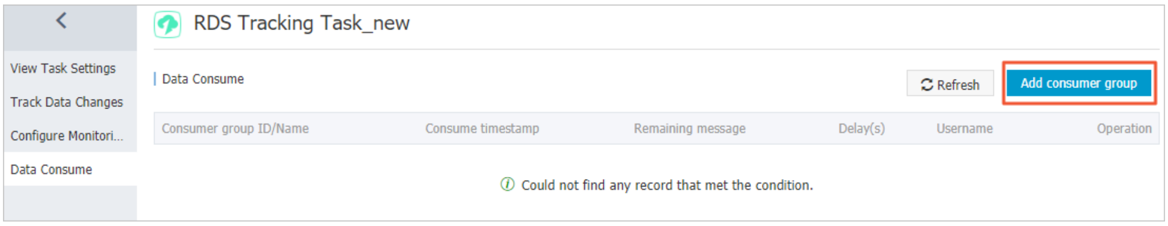
1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. In the upper part of the **Change Tracking Tasks** page, select the region where the change tracking instance resides.
4. Find the change tracking instance and click the instance ID.



□	Task ID/Name	Status	Consumption Checkpoint	Data Range	Billing Method	Actions
□	dts[redacted] New Subsc... RDS Tracking Task_new	Normal		2019-09-26 10:59:04 2019-09-26 11:25:51	Pay-As-You-Go	Switch to Subscription | View Tracked Data | Modify Required Objects | Add consumer group More

□ Delete Total: 1 item(s), Per Page: 20 item(s) « ‹ 1 › » GO

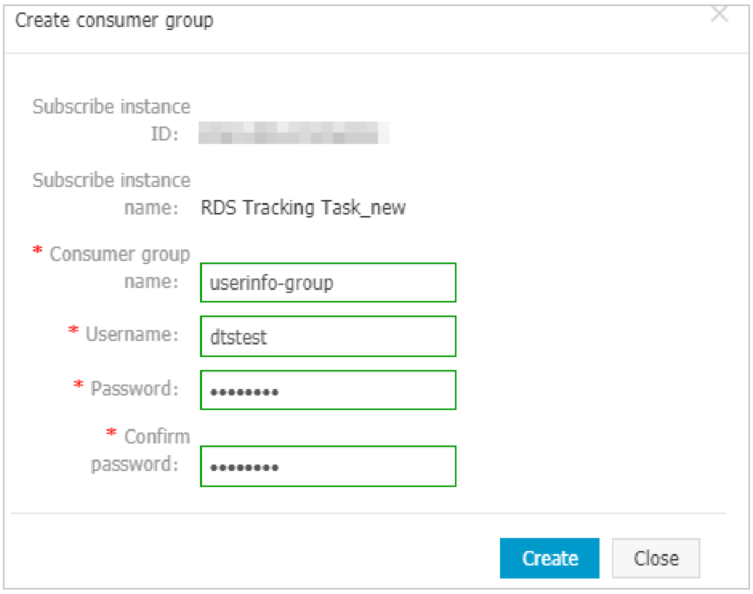
5. In the left-side navigation pane, click **Data Consume**.
6. On the **Data Consume** page, click **Add consumer group** in the upper-right corner.



Consumer group ID/Name	Consume timestamp	Remaining message	Delay(s)	Username	Operation

Could not find any record that met the condition.

7. In the **Create consumer group** dialog box, set parameters for the consumer group.



Create consumer group

Subscribe instance ID: [redacted]

Subscribe instance name: RDS Tracking Task_new

* Consumer group name: userinfo-group

* Username: dtstest

* Password: ••••••••

* Confirm password: ••••••••

Parameter	Description
Consumer group name	Enter a new name for the consumer group. We recommend that you use an informative name to identify the consumer group.
Username	Enter the username of the consumer group. ○ A username must contain one or more of the following character types: uppercase letters, lowercase letters, digits, and underscores (_). ○ A username must be 1 to 16 characters in length.
Password	Enter the password that corresponds to the username of the consumer group. ○ A password must contain two or more of the following characters: uppercase letters, lowercase letters, digits, and special characters. ○ A password must be 8 to 32 characters in length.
Confirm password	Enter the password again.

8. Click **Create**.

What to do next

After you create consumer groups, choose one of the following methods based on the client type to consume the tracked data:

- Use the SDK demo code to consume tracked data (recommended)
- Use flink-dts-connector to consume tracked data
- Use a Kafka client to consume tracked data

3.2. Manage consumer groups

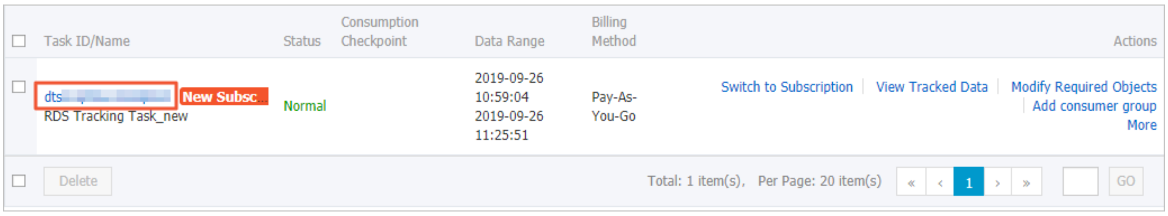
You can manage consumer groups of a change tracking instance in the DTS console. This topic describes how to modify the password of a consumer group and how to delete a consumer group.

Context

For more information about consumer groups, see Change tracking (new).

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. At the top of the **Change Tracking** page, select the region where your change tracking instance resides.
4. Find the change tracking instance and click the instance ID.



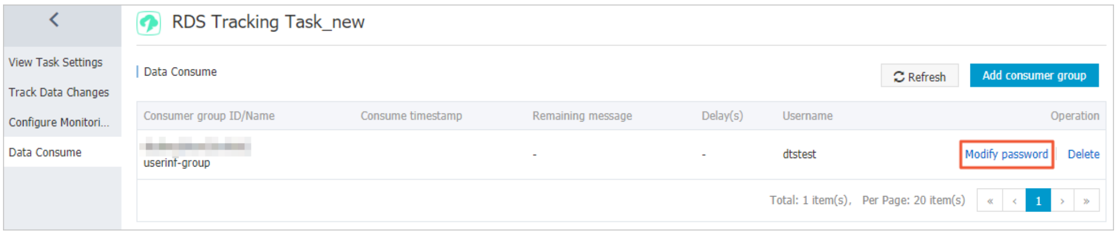
□	Task ID/Name	Status	Consumption Checkpoint	Data Range	Billing Method	Actions
□	dts[redacted] New Subsc... RDS Tracking Task_new	Normal		2019-09-26 10:59:04 2019-09-26 11:25:51	Pay-As-You-Go	Switch to Subscription | View Tracked Data | Modify Required Objects | Add consumer group More

□ Delete Total: 1 item(s), Per Page: 20 item(s) « ‹ 1 › » GO

5. In the left-side navigation pane, click **Consume Data**.
6. Modify the password of a consumer group or delete a consumer group.

   Modify the password of a consumer group

   i. On the **Consume Data** page, find the target consumer group and click **Modify Password** in the **Actions** column.



Consumer group ID/Name	Consume timestamp	Remaining message	Delay(s)	Username	Operation
[redacted] userinf-group		-	-	dtstest	Modify password Delete

Total: 1 item(s), Per Page: 20 item(s) « ‹ 1 › »

   ii. In the **Modify Password** dialog box that appears, enter the **old password** and **new password**, and enter the new password again in the **Confirm Password** field.

> Note
>
> - A password must contain two or more of the following character types: uppercase letters, lowercase letters, digits, and special characters.
> - The password must be 8 to 32 characters in length.

   iii. Click **Modify**.

   Delete a consumer group

> Note After a consumer group is deleted, the data in the group will be cleared and cannot be recovered. We recommend that you use caution when performing this operation.

   i. On the **Consume Data** page, find the target consumer group and click **Delete** in the **Actions** column.

   ii. In the **Delete Consumer Group** message that appears, click **OK**.

3.3. Use the SDK demo code to consume tracked data

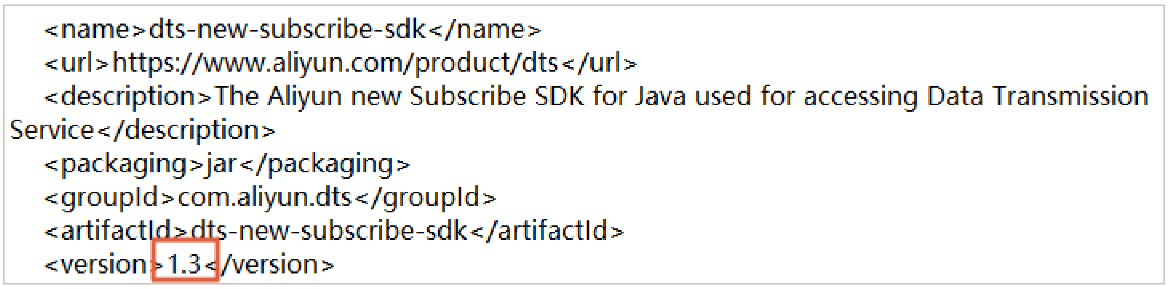
After you configure a change tracking task, you can use the SDK demo code that is provided by Data Transmission Service (DTS) to track and consume data. This topic describes how to use the SDK demo code to consume tracked data.

Procedure

> **Notice** If you use a RAM user to track and consume data, the RAM user must have the **AliyunDTSFullAccess** permission and the permissions to access the source objects. For more information, see Use a system policy to authorize a RAM user to manage DTS instances and Grant permissions to a RAM user.

IntelliJ IDEA (Community Edition 2020.1 Windows) is used in this example.

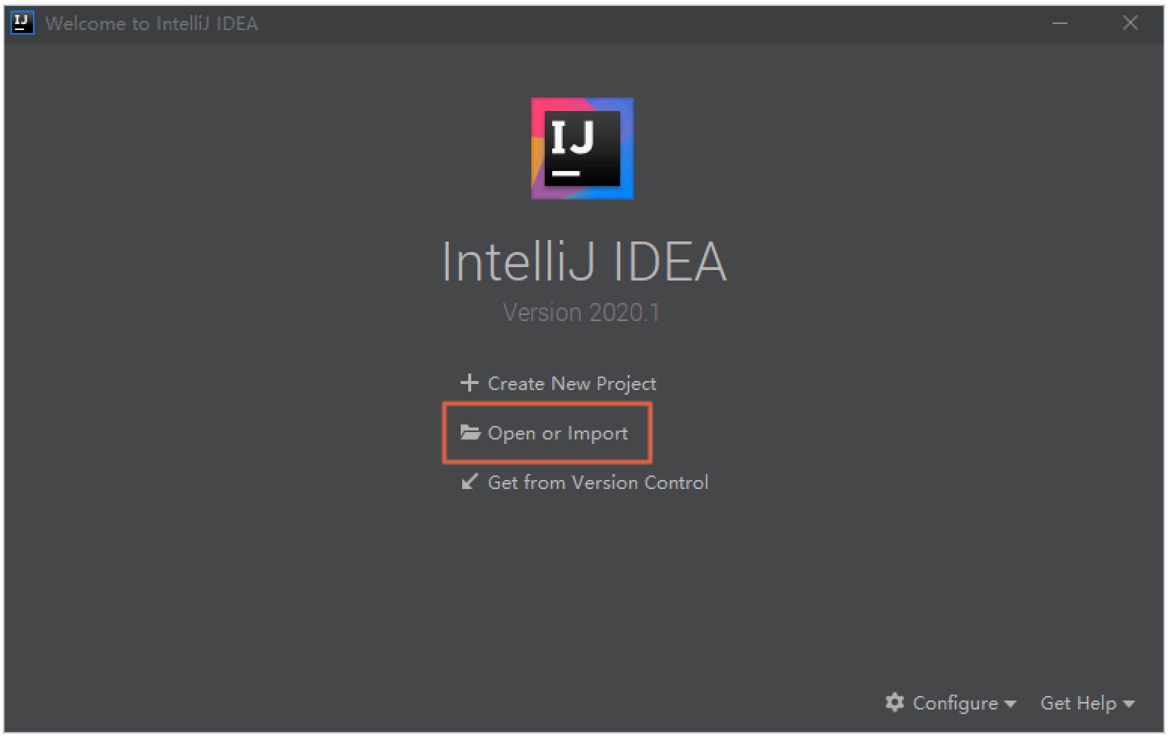
1. Create a change tracking task. For more information, see Track data changes from an ApsaraDB RDS for MySQL instance, Track data changes from a PolarDB for MySQL cluster, or Track data changes from a self-managed Oracle database.
2. Create one or more consumer groups. For more information, see Create consumer groups.
3. Download the SDK demo code package and decompress the package.
4. Go to the directory where the package is decompressed, open the *pom.xml* file by using a text editor, and then change the SDK version to the latest.



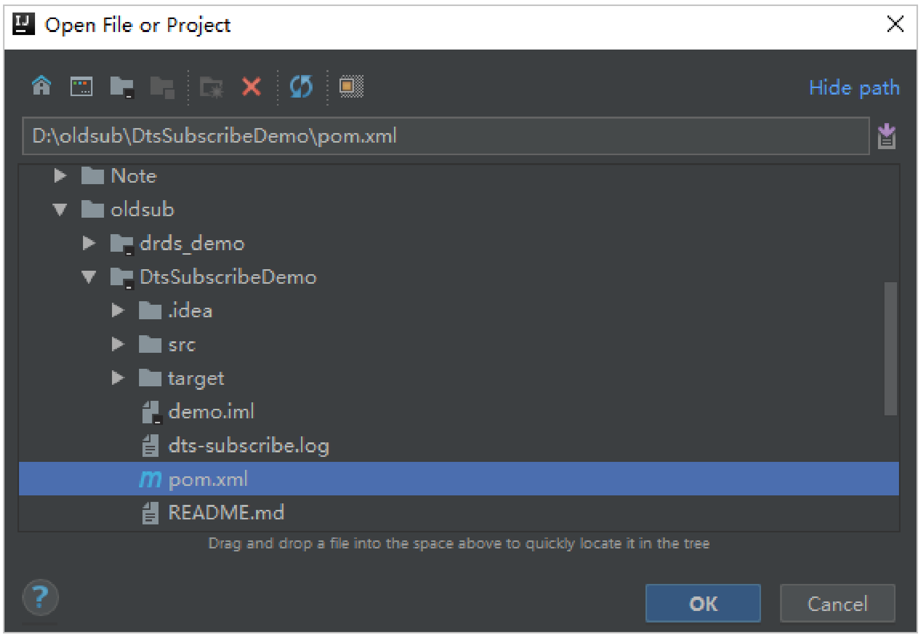
```
    <name>dts-new-subscribe-sdk</name>
    <url>https://www.aliyun.com/product/dts</url>
    <description>The Aliyun new Subscribe SDK for Java used for accessing Data Transmission Service</description>
    <packaging>jar</packaging>
    <groupId>com.aliyun.dts</groupId>
    <artifactId>dts-new-subscribe-sdk</artifactId>
    <version>1.3</version>
```

> **Notice** You can obtain the latest version of the change tracking SDK from the Maven website. For more information, visit the Maven page of the change tracking SDK.

5. Open IntelliJ IDEA. In the window that appears, click **Open or Import**.

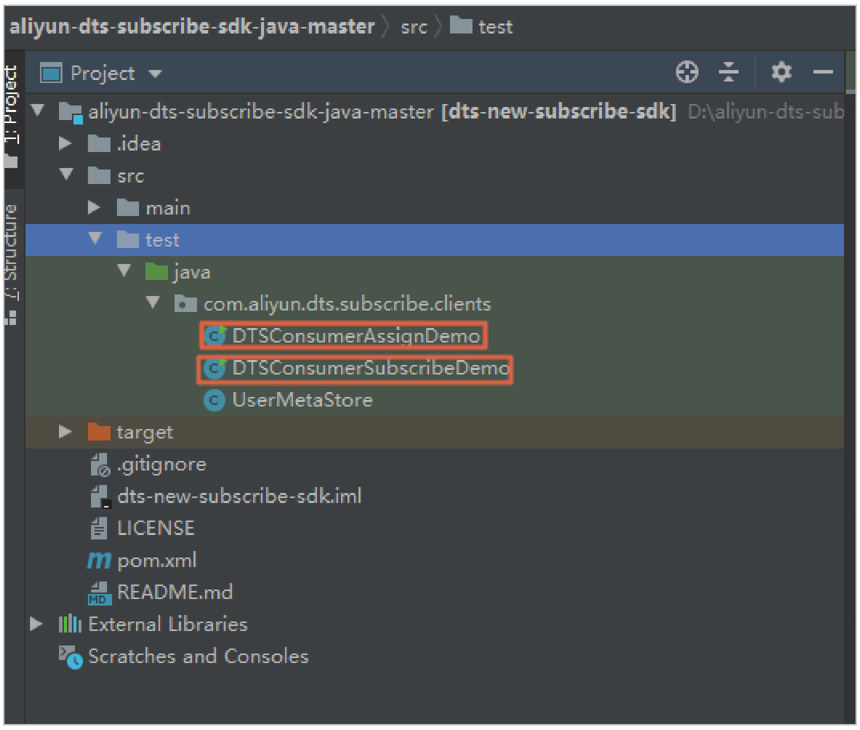


6. In the dialog box that appears, go to the directory where the package is decompressed, and open the folders to find the *pom.xml* file.



7. In the dialog box that is displayed, select **Open as Project**.

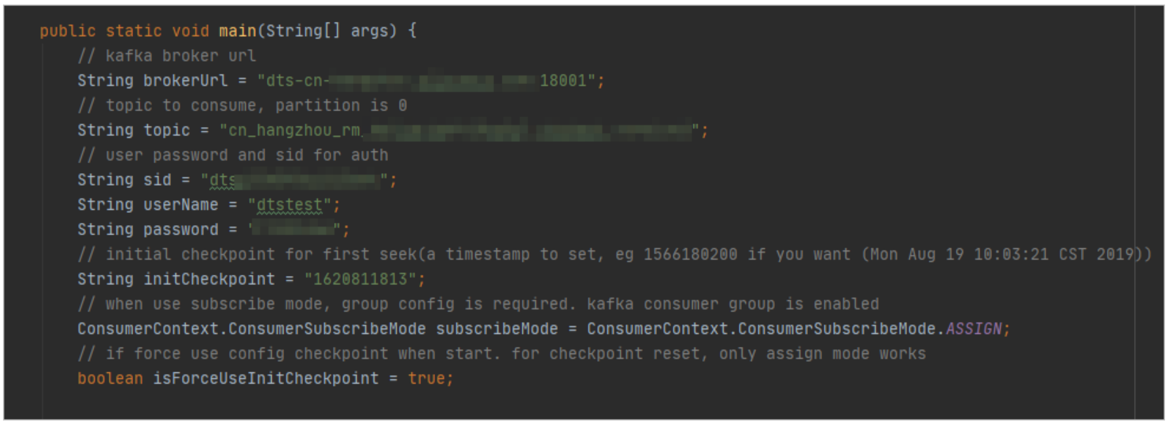
8. On the IntelliJ IDEA page, expand the folders to find the Java files. Then, double-click a Java file based on the mode in which you use the . The following Java files are available: *DTSConsumerAssignDemo.java* and *DTSConsumerSubscribeDemo.java*. SDK client



```
aliyun-dts-subscribe-sdk-java-master 〉 src 〉 test
Project
▼ aliyun-dts-subscribe-sdk-java-master [dts-new-subscribe-sdk] D:\aliyun-dts-sub
  ▶ .idea
  ▼ src
    ▶ main
    ▼ test
      ▼ java
        ▼ com.aliyun.dts.subscribe.clients
            DTSConsumerAssignDemo
            DTSConsumerSubscribeDemo
            UserMetaStore
  ▶ target
    .gitignore
    dts-new-subscribe-sdk.iml
    LICENSE
    pom.xml
    README.md
▶ External Libraries
  Scratches and Consoles
```

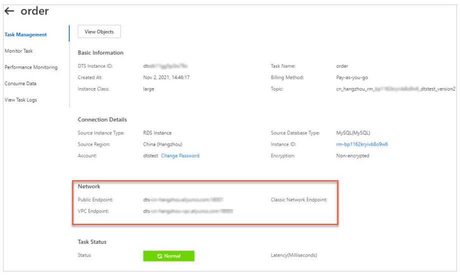
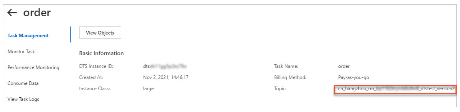
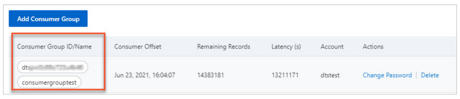
> **Note** DTS supports the following modes for using the SDK client:
>
> - ASSIGN mode: To ensure the global order of messages, DTS assigns only one partition (Partition 0) to each tracked topic. If you use the SDK client in ASSIGN mode, we recommend that you start only one SDK client.
> - SUBSCRIBE mode: To ensure the global order of messages, DTS assigns only one partition (Partition 0) to each tracked topic. In SUBSCRIBE mode, you can start multiple SDK clients in a consumer group at the same time to implement disaster recovery. If an SDK client in the consumer group fails, other SDK clients will be randomly and automatically allocated to Partition 0 to resume data consumption.

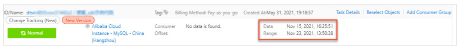
9. Set the required parameters in the code of the Java file.



```java
public static void main(String[] args) {
    // kafka broker url
    String brokerUrl = "dts-cn-[redacted]18001";
    // topic to consume, partition is 0
    String topic = "cn_hangzhou_rm[redacted]";
    // user password and sid for auth
    String sid = "dts[redacted]";
    String userName = "dtstest";
    String password = "[redacted]";
    // initial checkpoint for first seek(a timestamp to set, eg 1566180200 if you want (Mon Aug 19 10:03:21 CST 2019))
    String initCheckpoint = "1620811813";
    // when use subscribe mode, group config is required. kafka consumer group is enabled
    ConsumerContext.ConsumerSubscribeMode subscribeMode = ConsumerContext.ConsumerSubscribeMode.ASSIGN;
    // if force use config checkpoint when start. for checkpoint reset, only assign mode works
    boolean isForceUseInitCheckpoint = true;
```

Required parameters

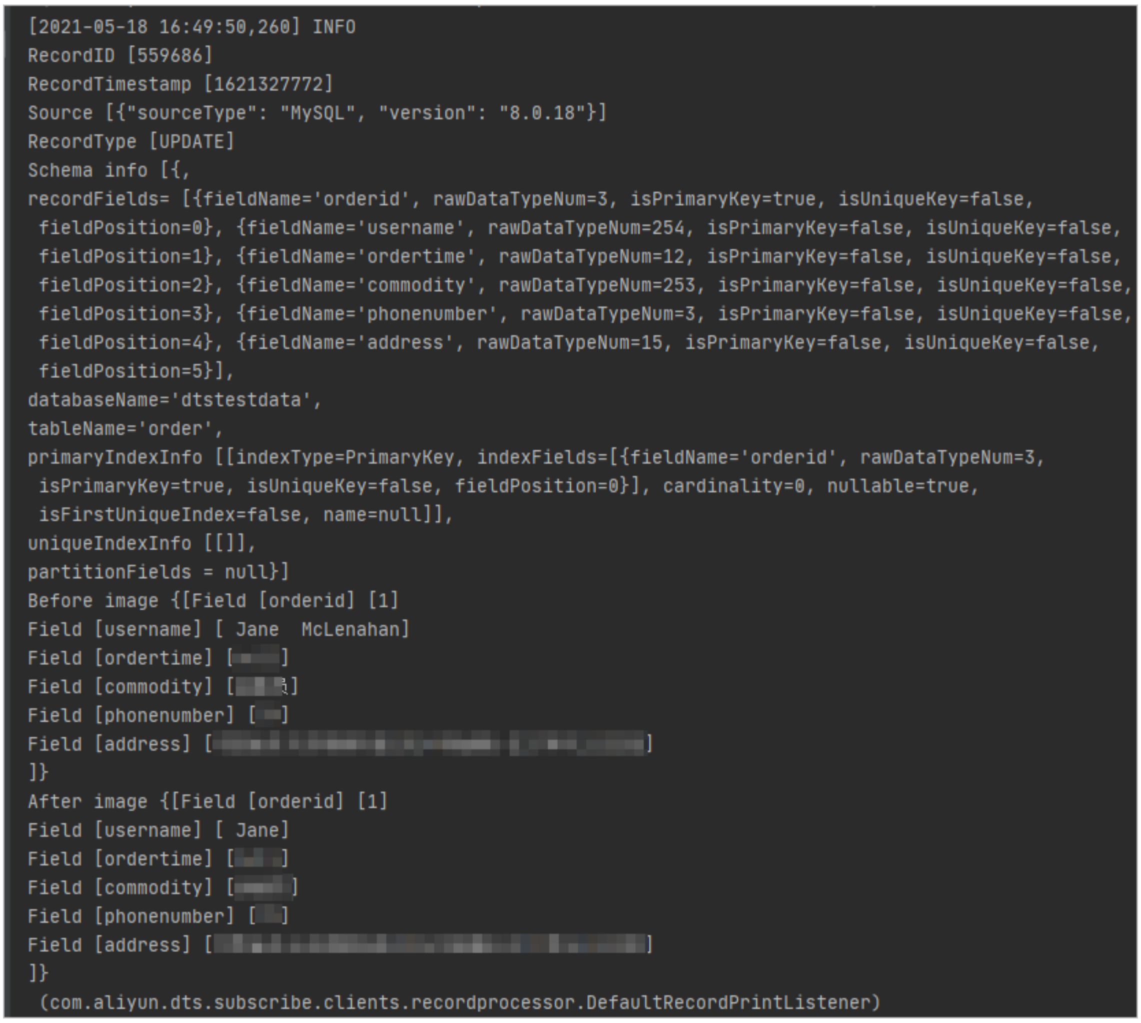
Parameter	Description	Method to obtain
`brokerUrl`	The network address and port number of the change tracking instance. ⓘ **Note** If you track data changes over internal networks, the network latency is minimal. This is applicable if the ECS instance where you deploy the SDK client belongs to the classic network or the same VPC as the change tracking instance.	In the DTS console, click the instance ID. On the **Task Management** page, you can obtain the network address and port number.  ← order Task Management Monitor Task Performance Monitoring Consume Data View Task Logs View Objects Basic Information DTS Instance ID: dts[redacted] Task Name: order Created At: Nov 2, 2021, 14:46:17 Billing Method: Pay-as-you-go Instance Class: large Topic: cn_hangzhou_rm_[redacted]_dtstest_version2 Connection Details Source Instance Type: RDS Instance Source Database Type: MySQL(MySQL) Source Region: China (Hangzhou) Instance ID: rm-bp1162kryivb8o9w6 Account: dtstest Change Password Encryption: Non-encrypted Network Public Endpoint: dts-[redacted] Classic Network Endpoint: VPC Endpoint: dts-[redacted] Task Status Status: Normal Latency(Milliseconds)
`topic`	The topic of the change tracking instance.	In the DTS console, click the instance ID. On the **Task Management** page, you can obtain the tracked **Topic**.  ← order Task Management Monitor Task Performance Monitoring Consume Data View Task Logs View Objects Basic Information DTS Instance ID: dts[redacted] Task Name: order Created At: Nov 2, 2021, 14:46:17 Billing Method: Pay-as-you-go Instance Class: large Topic: cn_hangzhou_rm_[redacted]_dtstest_version2
`sid`	The ID of the consumer group.	In the DTS console, click the instance ID, and then click **Consume Data**. You can obtain the **Consumer Group ID** and the corresponding **Account**. ⓘ **Note** The password of the consumer group account is automatically specified when you create a consumer group.  Add Consumer Group Consumer Group ID/Name: dts[redacted] consumergrouptest Consumer Offset: Jun 23, 2021, 16:04:07 Remaining Records: 14383181 Latency (s): 13211171 Account: dtstest Actions: Change Password | Delete
`userName`	The username of the consumer group. 🔔 **Warning** If you are not using the SDK client that is described in this topic, you must specify this parameter in the following format: `<Username>-<Consumer group ID>`, for example, `dtstest-dtsae******bpv`. Otherwise, the connection fails.	
`password`	The password of the account.	

Parameter	Description	Method to obtain
`initCheckpoint`	The consumer offset. It is the timestamp when the SDK client consumes the first data record. The value is a UNIX timestamp, for example, 1620962769. **Note** The consumer offset can be used in the following scenarios: - If the consumption process is interrupted, you can specify the consumer offset to resume data consumption. This allows you to prevent data loss. - When you start the change tracking client, you can specify the consumer offset to consume data on demand.	The consumer offset must be within the data range of the change tracking instance, as shown in the following figure. The consumer offset must be converted to a UNIX timestamp.  ID/Name: dts[redacted] Tag: Billing Method: Pay-as-you-go Created At:May 31, 2021, 19:18:57 Change Tracking (New) New Version Normal Alibaba Cloud Instance - MySQL - China (Hangzhou) Consumer Offset: No data is found. Data Range: Nov 15, 2021, 16:25:51 Nov 23, 2021, 13:50:38 **Note** You can use a search engine to obtain a UNIX timestamp converter.
`ConsumerContext.ConsumerSubscribeMode subscribeMode`	The mode in which you use the SDK client. Valid values: - `ConsumerContext.ConsumerSubscribeMode.ASSIGN`: In ASSIGN mode, only one SDK client in a consumer group can consume tracked data. For more information, see SDK usage modes. - `ConsumerContext.ConsumerSubscribeMode.SUBSCRIBE`: In SUBSCRIBE mode, you can start multiple SDK clients in a consumer group at the same time to implement disaster recovery. For more information, see SDK usage modes.	None

10. In the top menu bar of IntelliJ IDEA, choose **Run** > **Run** to run the client.

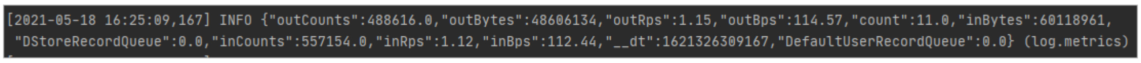
> Note When you run IntelliJ IDEA for the first time, it requires some time to load and install the relevant dependency.

- The following figure shows the result which indicates that the SDK client can track data changes from the source database.



```
[2021-05-18 16:49:50,260] INFO
RecordID [559686]
RecordTimestamp [1621327772]
Source [{"sourceType": "MySQL", "version": "8.0.18"}]
RecordType [UPDATE]
Schema info [{,
recordFields= [{fieldName='orderid', rawDataTypeNum=3, isPrimaryKey=true, isUniqueKey=false,
 fieldPosition=0}, {fieldName='username', rawDataTypeNum=254, isPrimaryKey=false, isUniqueKey=false,
 fieldPosition=1}, {fieldName='ordertime', rawDataTypeNum=12, isPrimaryKey=false, isUniqueKey=false,
 fieldPosition=2}, {fieldName='commodity', rawDataTypeNum=253, isPrimaryKey=false, isUniqueKey=false,
 fieldPosition=3}, {fieldName='phonenumber', rawDataTypeNum=3, isPrimaryKey=false, isUniqueKey=false,
 fieldPosition=4}, {fieldName='address', rawDataTypeNum=15, isPrimaryKey=false, isUniqueKey=false,
 fieldPosition=5}],
databaseName='dtstestdata',
tableName='order',
primaryIndexInfo [[indexType=PrimaryKey, indexFields=[{fieldName='orderid', rawDataTypeNum=3,
 isPrimaryKey=true, isUniqueKey=false, fieldPosition=0}], cardinality=0, nullable=true,
 isFirstUniqueIndex=false, name=null]],
uniqueIndexInfo [[]],
partitionFields = null}]
Before image {[Field [orderid] [1]
Field [username] [ Jane  McLenahan]
Field [ordertime] [[redacted]]
Field [commodity] [[redacted]]
Field [phonenumber] [[redacted]]
Field [address] [[redacted]]
]}
After image {[Field [orderid] [1]
Field [username] [ Jane]
Field [ordertime] [[redacted]]
Field [commodity] [[redacted]]
Field [phonenumber] [[redacted]]
Field [address] [[redacted]]
]}
 (com.aliyun.dts.subscribe.clients.recordprocessor.DefaultRecordPrintListener)
```

- The SDK client calculates and displays information about the consumed data at regular intervals. The information includes the total number of data records that are sent and received, the total amount of data, and the number of requests per second (RPS).



```
[2021-05-18 16:25:09,167] INFO {"outCounts":488616.0,"outBytes":48606134,"outRps":1.15,"outBps":114.57,"count":11.0,"inBytes":60118961,
 "DStoreRecordQueue":0.0,"inCounts":557154.0,"inRps":1.12,"inBps":112.44,"__dt":1621326309167,"DefaultUserRecordQueue":0.0} (log.metrics)
```

Statistics of consumed data

Parameter	Description
`outCounts`	The total number of data records consumed by the SDK client.
`outBytes`	The total amount of data consumed by the SDK client. Unit: bytes.
`outRps`	The number of requests per second when the SDK client consumes data.

Parameter	Description
`outBps`	The number of bits transmitted per second when the SDK client consumes data.
`inBytes`	The total amount of data that is sent by the DTS server. Unit: bytes.
`DStoreRecordQueue`	The size of the current data cache queue when the DTS server sends data.
`inCounts`	The total number of data records that are sent by the DTS server.
`inRps`	The number of requests per second when the DTS server sends data.
`__dt`	The current timestamp when the SDK client receives data. Unit: milliseconds.
`DefaultUserRecordQueue`	The size of the current data cache queue after serialization.

Save and query the consumer offset

When the SDK client is started for the first time or restarted, or an internal retry occurs, you need to query and specify the to start or resume data consumption. The following table describes how to manage and query the consumer offset in different situations. To implement on-demand data consumption, you must ensure that no data is lost or duplicate.

Scenario	SDK usage mode	Query method
Query the consumer offset	ASSIGN and SUBSCRIBE	• The SDK client saves the consumer offset every 5 seconds and submits it to the DTS server. To query the last consumer offset, you can use the following methods: ◦ Find the *localCheckpointStore* file of the server where the SDK client resides. ◦ Go to the **Consume Data** page of the change tracking instance. • If you configured an external persistent shared storage medium (such as a database) in `setUserRegisteredStore(newUserMetaStore())` in the *consumerContext.java* file, the storage medium saves the consumer offset every 5 seconds. You can query the consumer offset by using the storage medium.
When you start the SDK client for the first time, you must specify the consumer offset to consume data.	ASSIGN and SUBSCRIBE	Select the *DTSConsumerAssignDemo.java* or *DTSConsumerSubscribeDemo.java* file based on the mode in which you use the SDK client. Then, specify the `initCheckpoint` parameter. For more information, see Step 8 and Step 9.

Scenario	SDK usage mode	Query method
If an internal retry occurs, you must specify the consumer offset of the last data record to resume data consumption.	ASSIGN	Perform the following steps to find the consumer offset of the last data record: 1. Find the *localCheckpointStore* file of the server where the SDK client resides. 2. Find the consumer offset of the latest data record in IntelliJ IDEA. 3. Find the external storage medium that you configured in `setUserRegisteredStore(newUserMetaStore())` in the *consumerContext.java* file.
	SUBSCRIBE	Perform the following steps to find the consumer offset of the last data record: 1. Go to the **Consume Data** page of the change tracking instance and obtain the consumer offset. 2. Find the external storage medium that you configured in `setUserRegisteredStore(newUserMetaStore())` in the *consumerContext.java* file.
After the SDK client is restarted, you must specify the consumer offset of the last data record to resume data consumption.	ASSIGN	Check the setting of the `setForceUseCheckpoint` parameter in the *consumerContext.java* file and query the consumer offset. - If the parameter is set to `true`, the specified initCheckpoint is used as the consumer offset each time the SDK client is restarted. - If the parameter is set to `false` or is not specified, perform the following steps to find the consumer offset of the last data record: i. Go to the **Consume Data** page of the change tracking instance and obtain the consumer offset. ii. Find the external storage medium that you configured in `setUserRegisteredStore(newUserMetaStore())` in the *consumerContext.java* file. iii. Find the consumer offset of the latest data record in IntelliJ IDEA.
	SUBSCRIBE	In this mode, the setting of the `setForceUseCheckpoint` parameter in the *consumerContext.java* file does not take effect. Perform the following steps to find the consumer offset of the last data record: 1. Go to the **Consume Data** page of the change tracking instance and obtain the consumer offset. 2. Find the external storage medium that you configured in `setUserRegisteredStore(newUserMetaStore())` in the *consumerContext.java* file. 3. Find the consumer offset of the latest data record in IntelliJ IDEA.

consumer offset

Troubleshooting

Issue	Error message	Cause	Solution
Connection failed	`ERROR CheckResult{isOk=false, errMsg='telnet dts-cn-hangzhou.aliyuncs.com:18009 failed, please check the network and if the brokerUrl is correct'} (com.aliyun.dts.subscribe.clients.DefaultDTSConsumer)`	The specified `brokerUrl` is invalid.	Enter the valid `brokerUrl` , `userName` , and `password` . For more information, see Required parameters.
	`telnet real node *** failed, please check the network`	The broker address cannot be redirected to the real IP address.	
	`ERROR CheckResult{isOk=false, errMsg='build kafka consumer failed, error: org.apache.kafka.common.errors.TimeoutException: Timeout expired while fetching topic metadata, probably the user name or password is wrong'} (com.aliyun.dts.subscribe.clients.DefaultDTSConsumer)`	The specified username and password are invalid.	
	`com.aliyun.dts.subscribe.clients.exception.TimestampSeekException: RecordGenerator:seek timestamp for topic [cn_hangzhou_rm_bp11tv2923n87081s_rdsdt_dtsacct-0] with timestamp [1610249501] failed`	The `setUseCheckpoint` parameter in the *consumerContext.java* file is set to `true` , but the consumer offset is not within the data range of the change tracking instance.	Enter the consumer offset within the data range of the change tracking instance. For more information, see Required parameters.

Issue	Error message	Cause	Solution
The response time of data consumption is increased.	None	You can analyze the cause by querying the DStoreRecordQueue and DefaultUserRecordQueue parameters. For more information, see Statistics of consumed data. ● If the DStoreRecordQueue parameter is set to 0, the DTS server pulls data at a slower rate. ● If the DefaultUserRecordQueue parameter is set to the default value 512, the SDK client consumes data at a slower rate.	

3.4. Use the SDK demo code to consume the tracked data from a PolarDB-X 1.0 instance

After you configure a change tracking task, you can use the SDK demo code that is provided by Data Transmission Service (DTS) to consume the tracked data. This topic describes how to use the SDK demo code to consume tracked data from distributed databases such as PloarDB-X 1.0 (or ApsaraDB RDS for MySQL) and Data Management (DMS) LogicDB (or PolarDB for MySQL).

Prerequisites

- The Java Development Kit (JDK) of the 1.8 version is installed.
- IntelliJ IDEA is installed.

Precautions

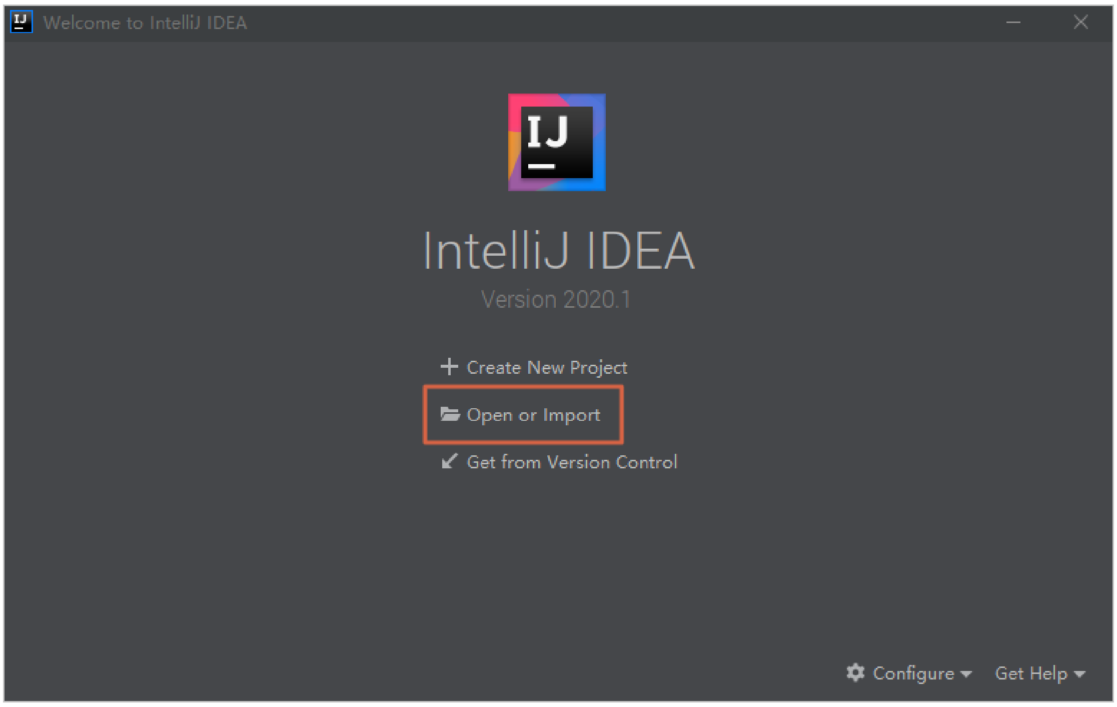
If you use a Resource Access Management (RAM) user to track data, the RAM user must have the **AliyunDTSFullAccess** permission and permissions to access the source objects. For more information, see Use a system policy to authorize a RAM user to manage DTS instances and Grant permissions to a RAM user.

Procedure

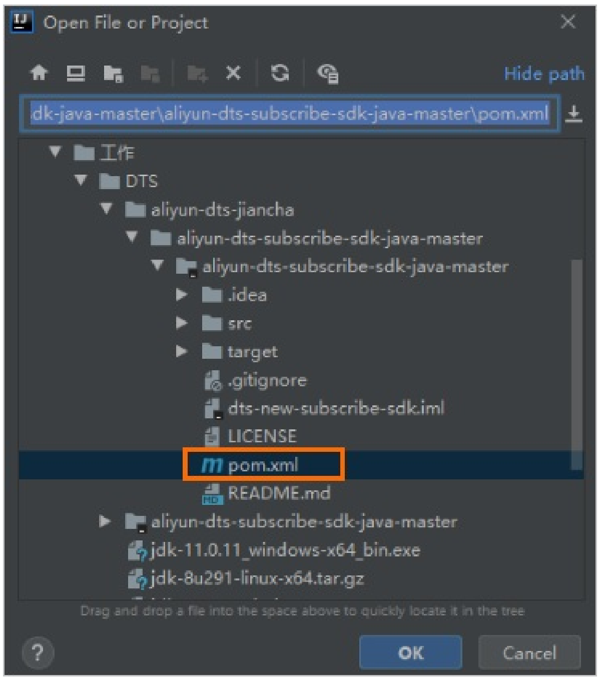
This topic describes how to use the SDK demo code to consume the tracked data from a PolarDB-X 1.0 instance. IntelliJ IDEA (Community Edition 2020.1 for Windows) is used in this example.

1. Create a change tracking instance. For more information, see Track data changes from a PolarDB-X 1.0PolarDB-X 1.0 instance.
2. Create one or more consumer groups. For more information, see Create consumer groups.
3. Download the SDK demo code package and decompress the package.
4. Open the destination project in IntelliJ IDEA.

i. Open IntelliJ IDEA. In the window that appears, click **Open or Import**.



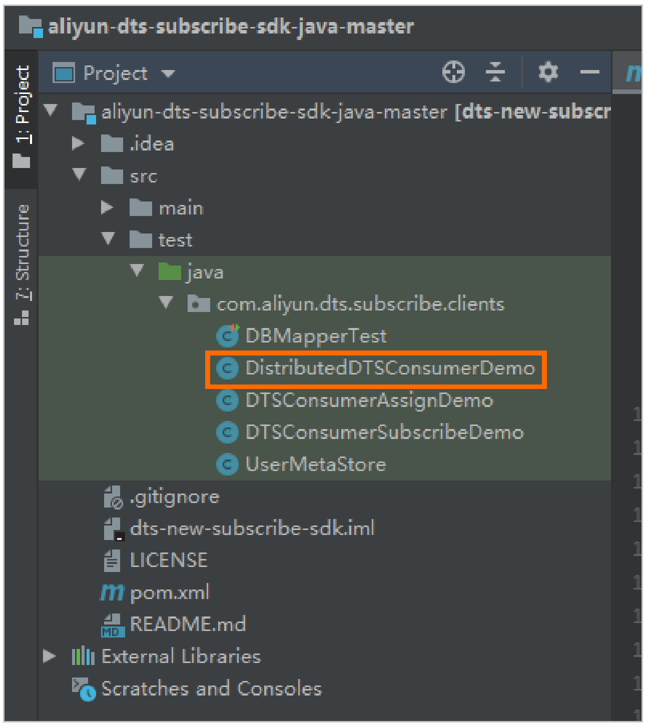
ii. In the dialog box that is displayed, go to the directory where the package is decompressed. Then, open the folders and double-click the *pom.xml* file.


Open File or Project
Hide path
工作
DTS
aliyun-dts-jiancha
aliyun-dts-subscribe-sdk-java-master
aliyun-dts-subscribe-sdk-java-master
.idea
src
target
.gitignore
dts-new-subscribe-sdk.iml
LICENSE
pom.xml
README.md
aliyun-dts-subscribe-sdk-java-master
jdk-11.0.11_windows-x64_bin.exe
jdk-8u291-linux-x64.tar.gz
Drag and drop a file into the space above to quickly locate it in the tree
?
OK
Cancel

dk-java-master\aliyun-dts-subscribe-sdk-java-master\pom.xml

iii. In the dialog box that appears, select **Open as Project**.

5. In the IntelliJ IDEA window, expand the folders to find the Java files. Then, double-click a Java file based on the mode in which you use the In this scenario, select *DistributedDTSConsumerDemo*. SDK client



6. Set the required parameters in the code of the Java file.

```
public static void main(String[] args) throws ClientException {
        // Configure a change tracking task for a distributed database such as a PolarD
B-X 1.0 instance. Specify parameters related to your AccessKey, DTS instance, DTS job,
and consumer groups.
        String accessKeyId = "LTA***********99reZ";
        String accessKeySecret = "****************";
        String regionId = "cn-hangzhou";
        String dtsInstanceId = "dtse5212sed162****";
        String jobId = "l791216x16d****";
        String sid = "dtsip412t13160****";
        String userName = "xftest";
        String password = "******";
        String proxyUrl = "dts-cn-****.com:18001";
        // initial checkpoint for first seek(a timestamp to set, eg 1566180200 if you w
ant (Mon Aug 19 10:03:21 CST 2019))
        String checkpoint = "1639620090";
        // Convert physical database/table name to logical database/table name
        boolean mapping = true;
        // if force use config checkpoint when start. for checkpoint reset, only assign
mode works
        boolean isForceUseInitCheckpoint = false;
        ConsumerContext.ConsumerSubscribeMode subscribeMode = ConsumerContext.ConsumerS
ubscribeMode.ASSIGN;
        DistributedDTSConsumerDemo demo = new DistributedDTSConsumerDemo(userName, pass
word, regionId,
                jobId, sid, dtsInstanceId, accessKeyId, accessKeySecret, subscribeMode,
proxyUrl,
                checkpoint, isForceUseInitCheckpoint, mapping);
        demo.start();
    }
```

Parameter	Description	Method to obtain
accessKeyId	The AccessKey ID.	For more information about how to obtain an AccessKey ID, see Create and obtain an AccessKey pair.
accessKeySecret	The AccessKey secret.	
regionId	The ID of the region where the change tracking instance resides.	In the new DTS console, click the instance ID. On the **Task Management** page, you can obtain the instance region information. For example, if the instance resides in the China (Hangzhou) region, set the parameter to `cn-hangzhou`. For more information, see List of supported regions.
dtsInstanceId	The ID of the change tracking instance.	In the new DTS console, click the instance ID. On the **Task Management** page, you can obtain the instance ID and task ID.
jobId	The ID of the change tracking task.	
sid	The ID of the consumer group.	In the new DTS console, click the instance ID. In the left-side navigation pane, click **Consume Data**. You can obtain the **Sid** and the corresponding **Account**. **Note** The password of the consumer group account is automatically specified when you create a consumer group.
userName	The account of the consumer group.	
password	The password that corresponds to the account of the consumer group.	
proxyUrl	The endpoint and port number of the change tracking instance. **Note** If you track data changes over internal networks, the network latency is minimal. This is applicable if the ECS instance where you deploy the SDK client belongs to the classic network or the same virtual private cloud (VPC) as the change tracking instance.	In the new DTS console, click the instance ID. On the **Task Management** page, you can obtain the endpoint and port number.

Parameter	Description	Method to obtain
checkpoint	The consumer offset. It is the timestamp when the SDK client consumes the first data record. The value is a UNIX timestamp in seconds. Note The consumer offset can be used in the following scenarios: - After the consumption process is interrupted, you can specify the consumer offset to resume data consumption. This allows you to prevent data loss. - When you start the change tracking client, you can specify the consumer offset to consume data on demand.	The consumer offset must be within the data range of the change tracking instance. The consumer offset must be converted into a UNIX timestamp. Note You can use a search engine to obtain a UNIX timestamp converter.

7. In the top menu bar of IntelliJ IDEA, choose **Run > Run** to run the client.

> **Note** When you run IntelliJ IDEA for the first time, it requires some time to load and install the relevant dependency.

- The running result shows the result that the SDK client can track data changes from the source instance.
- The SDK client collects and displays statistics about the consumed data at regular intervals. The statistics information includes the total number of data records that are sent and received, the total amount of data, and the number of requests per second (RPS). The following table describes the parameters in the information.

Parameter	Description
`outCounts`	The total number of data records consumed by the SDK client.
`outBytes`	The total amount of data consumed by the SDK client. Unit: bytes.
`outRps`	The number of RPS when the SDK client consumes data.
`outBps`	The number of bits transmitted per second when the SDK client consumes data.
`count`	None
`inBytes`	The total amount of data that is sent by the DTS server. Unit: bytes.

Parameter	Description
`DStoreRecordQueue`	The size of the current data cache queue when the DTS server sends data.
`inCounts`	The total number of data records that are sent by the DTS server.
`inRps`	The number of RPS when the DTS server sends data.
`inBps`	The number of bits transmitted per second when the DTS server sends data.
`__dt`	The current timestamp when the SDK client receives data. Unit: milliseconds.
`DefaultUserRecordQueue`	The size of the current data cache queue after serialization.

8. (Optional) To modify the data type of the data to track, you can modify the code in the `buildRecordListener()` method or use a custom class.

```
public static Map<String, RecordListener> buildRecordListener() {
        // user can impl their own listener
        RecordListener mysqlRecordPrintListener = new RecordListener() {
            @Override
            public void consume(DefaultUserRecord record) {
                OperationType operationType = record.getOperationType();
                if (operationType.equals(OperationType.INSERT)
                        || operationType.equals(OperationType.UPDATE)
                        || operationType.equals(OperationType.DELETE)
                        || operationType.equals(OperationType.HEARTBEAT)) {
                    // consume record
                    RecordListener recordPrintListener = new DefaultRecordPrintListener
(DbType.MySQL);
                    recordPrintListener.consume(record);
                    //commit method push the checkpoint update
                    record.commit("");
                }
            }
        };
        return Collections.singletonMap("mysqlRecordPrinter", mysqlRecordPrintListener)
;
    }
```

3.5. Use flink-dts-connector to consume tracked data

After you configure a change tracking task, you can use the flink-dts-connector file to consume tracked data. This topic describes how to use the flink-dts-connector file to consume tracked data.

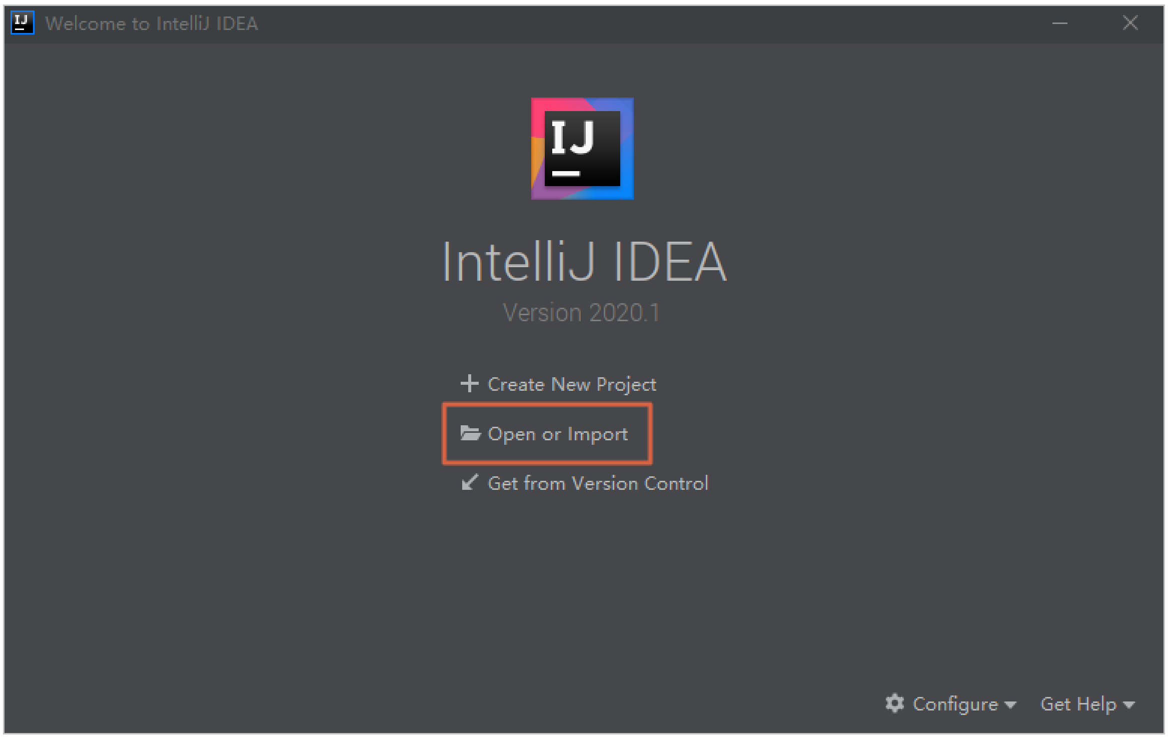
Limits

- Data Transmission Service (DTS) supports the following types of Flink programs: DataStream API and Table API & SQL.
- If you use a Table API & SQL program, you can consume the data of only one table each time you configure a change tracking task. If you want to consume the data of multiple tables, you must configure a task for each table.

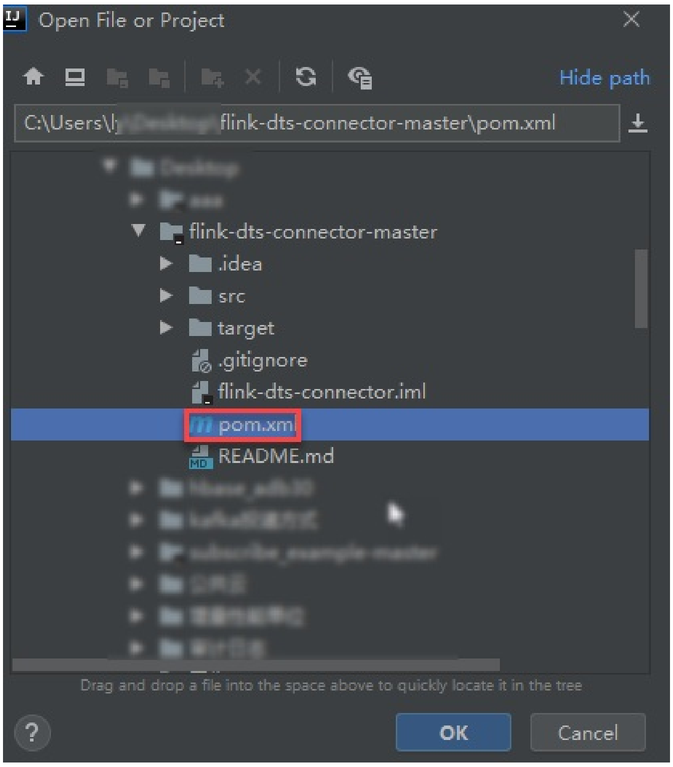
Procedure

IntelliJ IDEA (Community Edition 2020.1 Windows) is used in this example.

1. Create a change tracking task. For more information, see Track data changes from an ApsaraDB RDS for MySQL instance, Track data changes from a PolarDB for MySQL cluster, or Track data changes from a self-managed Oracle database.
2. Create one or more consumer groups. For more information, see Create consumer groups.
3. Download the flink-dts-connector file and decompress it.
4. Open IntelliJ IDEA. In the window that appears, click **Open or Import**.



5. In the dialog box that appears, go to the directory where the flink-dts-connector file is decompressed, and expand the folders to find the *pom.xml* file.

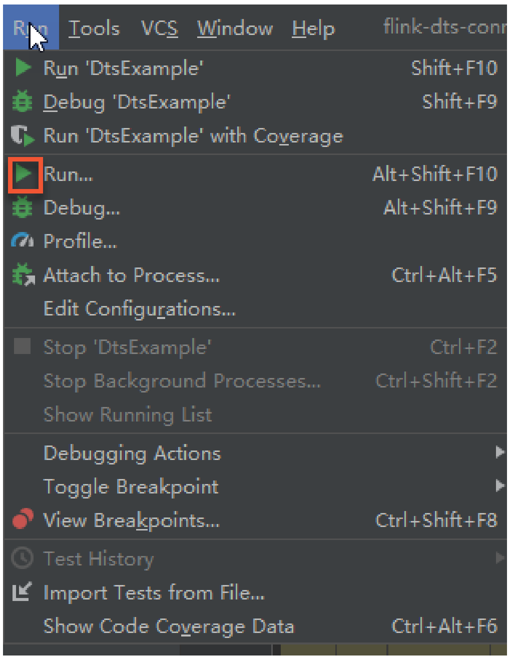

Open File or Project
Hide path
C:\Users\l[redacted]flink-dts-connector-master\pom.xml
[redacted]
flink-dts-connector-master
.idea
src
target
.gitignore
flink-dts-connector.iml
pom.xml
README.md
[redacted]
Drag and drop a file into the space above to quickly locate it in the tree
?
OK
Cancel

6. In the dialog box that is displayed, select **Open as Project**.
7. Add the following dependency to the *pom.xml* file:

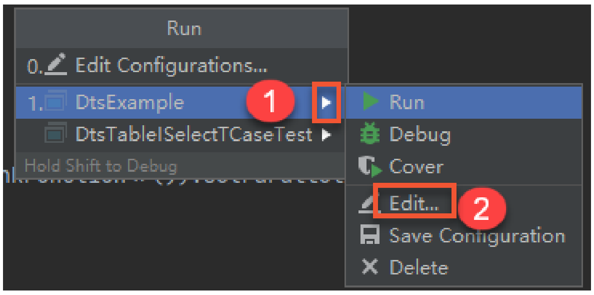
```
<dependency>
      <groupId>com.alibaba.flink</groupId>
      <artifactId>flink-dts-connector</artifactId>
      <version>1.1.1-SNAPSHOT</version>
      <classifier>jar-with-dependencies</classifier>
</dependency>
```

8. On the IntelliJ IDEA page, expand the folders to find the Java files. Then, double-click a Java file based on the type of Flink connector that you use.
   - If you use a DataStream API connector, you must double-click the *flink-dts-connector-master\src\test\java\com\alibaba\flink\connectors\dts\datastream\DtsExample.java* file and perform the following operations:

a. In the top menu bar of IntelliJ IDEA, click the Run icon.


Run
Tools
VCS
Window
Help
flink-dts-conn
Run 'DtsExample' Shift+F10
Debug 'DtsExample' Shift+F9
Run 'DtsExample' with Coverage
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Profile...
Attach to Process... Ctrl+Alt+F5
Edit Configurations...
Stop 'DtsExample' Ctrl+F2
Stop Background Processes... Ctrl+Shift+F2
Show Running List
Debugging Actions
Toggle Breakpoint
View Breakpoints... Ctrl+Shift+F8
Test History
Import Tests from File...
Show Code Coverage Data Ctrl+Alt+F6

b. In the dialog box that appears, choose **DtsExample > Edit**.

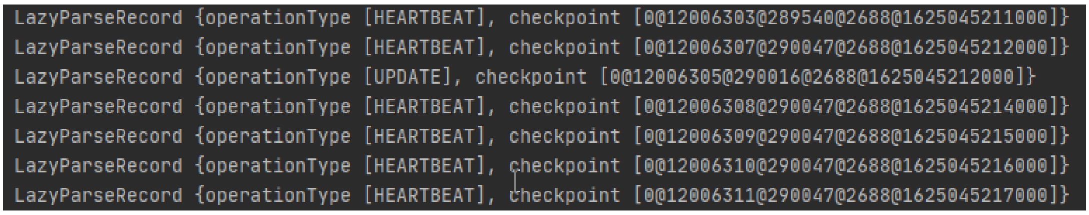


c. In the **Program arguments** field, enter the parameters and corresponding values, and then click **Run** to run flink-dts-connector.

> **Note** For more information about the parameters and query methods, see Parameters.

```
--broker-url dts-cn-******.******.***:****** --topic cn_hangzhou_rm_**********_dt
stest_version2 --sid dts****** --user dtstest --password Test123456 --checkpoint
1624440043
```

d. The following figure shows that the Flink program can track data changes from the source database.



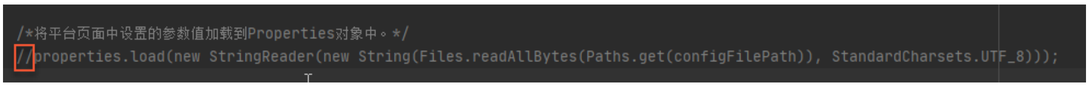
```
LazyParseRecord {operationType [HEARTBEAT], checkpoint [0@12006303@289540@2688@1625045211000]}
LazyParseRecord {operationType [HEARTBEAT], checkpoint [0@12006307@290047@2688@1625045212000]}
LazyParseRecord {operationType [UPDATE], checkpoint [0@12006305@290016@2688@1625045212000]}
LazyParseRecord {operationType [HEARTBEAT], checkpoint [0@12006308@290047@2688@1625045214000]}
LazyParseRecord {operationType [HEARTBEAT], checkpoint [0@12006309@290047@2688@1625045215000]}
LazyParseRecord {operationType [HEARTBEAT], checkpoint [0@12006310@290047@2688@1625045216000]}
LazyParseRecord {operationType [HEARTBEAT], checkpoint [0@12006311@290047@2688@1625045217000]}
```

> **Note** To query specific records of data changes, you can go to the Task Manager page of the Flink program.

- If you use a Table API & SQL connector, you must double-click the *flink-dts-connector-master\src\test\java\com\alibaba\flink\connectors\dts\sql\DtsTableISelectTCaseTest.java* file and perform the following operations:

> **Note** A single `DtsTableISelectTCaseTest.java` file can be used to configure only one change tracking task and consume the data of only one table. If you want to consume the data of multiple tables, you must configure a task for each table.

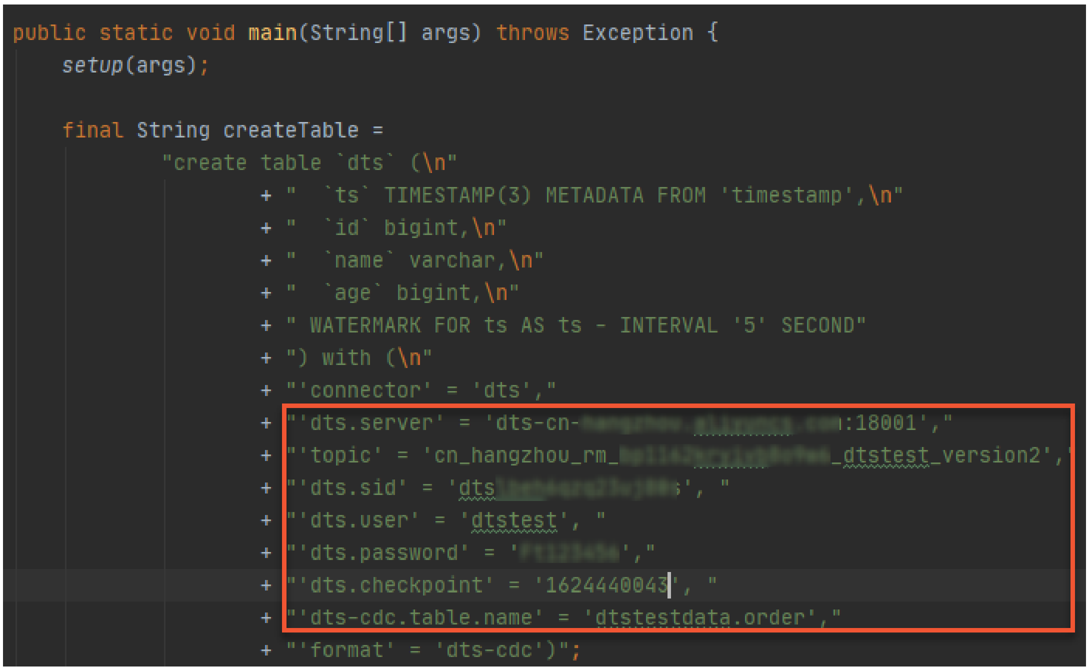
a. Enter two forward slashes (`//`) and add comments, as shown in the following figure.



```
/*将平台页面中设置的参数值加载到Properties对象中。*/
//properties.load(new StringReader(new String(Files.readAllBytes(Paths.get(configFilePath)), StandardCharsets.UTF_8)));
```

b. Specify the information of the table from which you want to track data changes. SQL statements are supported.

c. Set the parameters required for the change tracking instance. For more information, see Parameters.

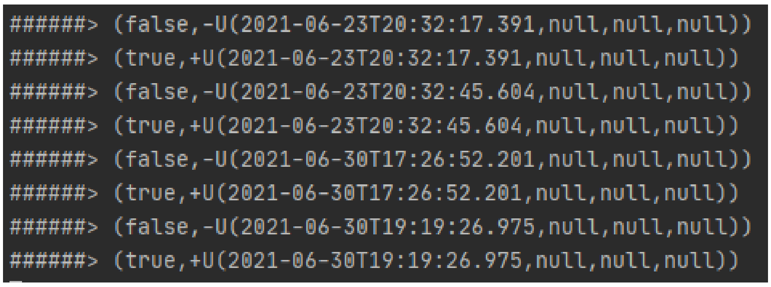


```
public static void main(String[] args) throws Exception {
    setup(args);

    final String createTable =
            "create table `dts` (\n"
                    + "  `ts` TIMESTAMP(3) METADATA FROM 'timestamp',\n"
                    + "  `id` bigint,\n"
                    + "  `name` varchar,\n"
                    + "  `age` bigint,\n"
                    + " WATERMARK FOR ts AS ts - INTERVAL '5' SECOND"
                    + ") with (\n"
                    + "'connector' = 'dts',"
                    + "'dts.server' = 'dts-cn-[redacted]:18001',"
                    + "'topic' = 'cn_hangzhou_rm_[redacted]_dtstest_version2',"
                    + "'dts.sid' = 'dts[redacted]', "
                    + "'dts.user' = 'dtstest', "
                    + "'dts.password' = '[redacted]',"
                    + "'dts.checkpoint' = '1624440043', "
                    + "'dts-cdc.table.name' = 'dtstestdata.order',"
                    + "'format' = 'dts-cdc')";
```

d. In the top menu bar of IntelliJ IDEA, click **Run'DtsTableISelectTCaseTest'** to run flink-dts-connector.

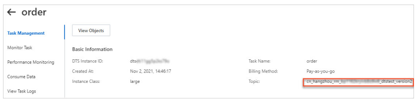
e. The following figure shows that the Flink program can track data changes from the source database.

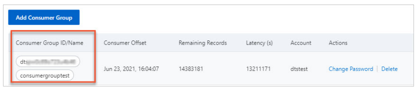
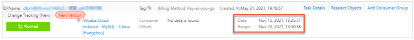


```
######> (false,-U(2021-06-23T20:32:17.391,null,null,null))
######> (true,+U(2021-06-23T20:32:17.391,null,null,null))
######> (false,-U(2021-06-23T20:32:45.604,null,null,null))
######> (true,+U(2021-06-23T20:32:45.604,null,null,null))
######> (false,-U(2021-06-30T17:26:52.201,null,null,null))
######> (true,+U(2021-06-30T17:26:52.201,null,null,null))
######> (false,-U(2021-06-30T19:19:26.975,null,null,null))
######> (true,+U(2021-06-30T19:19:26.975,null,null,null))
```

> **Note** To query specific records of data changes, you can go to the Task Manager page of the Flink program.

Parameters

Parameters in the DstExample file	Parameters in the DtsTableISelectTCaseTest file	Description	Query method
`broker-url`	`dts.server`	The network address and port number of the change tracking instance. **Note** If you track data changes over internal networks, the network latency is minimal. This is applicable if the Elastic Compute Service (ECS) instance where you deploy the Flink program belongs to the classic network or the same VPC as the change tracking instance.	In the DTS console, click the instance ID. On the **View Task Settings** page, you can obtain the **tracked topic**, network address, and port number.  ← order Task Management Monitor Task Performance Monitoring Consume Data View Task Logs View Objects Basic Information DTS Instance ID: dts[redacted] Created At: Nov 2, 2021, 14:46:17 Instance Class: large Task Name: order Billing Method: Pay-as-you-go Topic: cn_hangzhou_rm_[redacted]_dtstest_version2
`topic`	`topic`	The topic of the change tracking instance.	

Parameters in the DstExample file	Parameters in the DtsTableISelectTCaseTest file	Description	Query method
`sid`	`dts.sid`	The ID of the consumer group.	In the DTS console, click the instance ID, and then click **Data Consume**. You can obtain the **consumer group ID** and the corresponding **username**. **Note** When you create a consumer group, the password of the consumer group is automatically specified.  Add Consumer Group Consumer Group ID/Name | Consumer Offset | Remaining Records | Latency (s) | Account | Actions dts[redacted] consumergrouptest | Jun 23, 2021, 16:04:07 | 14383181 | 13211171 | dtstest | Change Password | Delete
`user`	`dts.user`	The username of the consumer group. **Warning** If you are not using the flink-dts-connector file that is described in this topic, you must specify this parameter in the following format: `<Username>-<Consumer group ID>` , for example, `dtstest-dtsae******bpv` . Otherwise, the connection fails.	
`password`	`dts.password`	The password of the consumer group.	
`checkpoint`	`dts.checkpoint`	The consumer offset. It is the timestamp when flink-dts-connector consumes the first data record. The value is a UNIX timestamp, for example, 1624440043. **Note** The consumer offset is useful in the following scenarios: - If the consumption process is interrupted, you can specify the consumer offset on the change tracking client to resume data consumption. This allows you to prevent against data loss. - When you start the change tracking client, you can specify the consumer offset to consume data on demand.	The consumer offset must be within the data range of the change tracking instance, as shown in the following figure. The consumer offset must be converted to a UNIX timestamp.  ID/Name: dts[redacted] Tag: Billing Method: Pay-as-you-go Created At:May 31, 2021, 19:18:57 Task Details | Reselect Objects | Add Consumer Group Change Tracking (New) New Version Normal Alibaba Cloud Instance - MySQL - China (Hangzhou) Consumer Offset: No data is found. Data Range: Nov 15, 2021, 16:25:51 Nov 23, 2021, 13:50:38 **Note** You can use a search engine to obtain a UNIX timestamp converter.

Parameters in the DstExample file	Parameters in the DtsTableISelectTCaseTest file	Description	Query method
None	`dts.cdc.table.name`	The objects for change tracking. You can specify only a single table in the format of `<Database name>.<Table name>`, for example, `dtstestdata.order`.	In the DTS console, click the instance ID. On the **View Task Settings** page, click **View Objects** to view the database name and table name.

FAQ

Error message	Possible cause	Solution
`Cluster changed from *** to ***, consumer require restart.`	The DStore module used by DTS to read incremental data is switched. As a result, the consumer offset of the Flink program is lost.	You do not need to restart the Flink program. You only need to query the consumer offset of the Flink program and set the `checkpoint` or `dts.checkpoint` parameter in the *DtsExample.java* and *DtsTableISelectTCaseTest.java* files again to resume data consumption.

3.6. Use a Kafka client to consume tracked data

This topic describes how to use the demo code of a Kafka client to consume tracked data. The change tracking feature of the new version allows you to consume tracked data by using a Kafka client (V0.11 to V2.0).

Precautions

- If you enable auto commit when you use the change tracking feature, some data may be committed before it is consumed. This results in data loss. We recommend that you manually commit data.

  > **Note** If data fails to be committed, you can restart the client to continue consuming data from the last recorded consumer offset. However, duplicate data may be generated during this period. You must manually filter out the duplicate data.

- Data is serialized and stored in the Avro format. For more information, see Record.avsc.

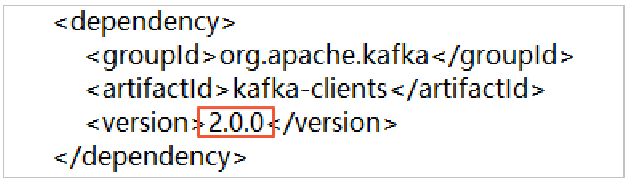
  > **Warning** If you are not using the Kafka client that is described in this topic, you must parse the tracked data based on the Avro schema.

- The search unit is second when Data Transmission Service (DTS) calls the `offsetForTimes` operation. The search unit is millisecond when a native Kafka client calls this operation.

Run the Kafka client

Click Kafka client demo code to download the demo code of the Kafka client. For more information about how to use the demo code, see Readme.

> **Note** If you use a Kafka client of version 2.0, you must change the version number in the *subscribe_example-master/javaimpl/pom.xml* file to 2.0.0.



```
<dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka-clients</artifactId>
    <version>2.0.0</version>
</dependency>
```

Steps

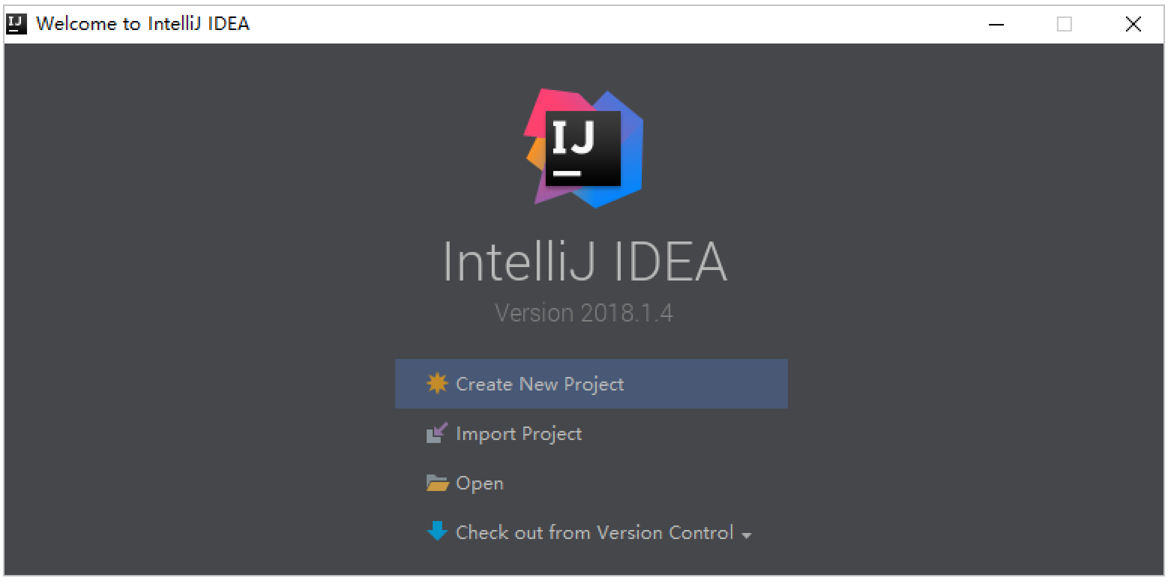
Step	File or directory
1. Use the native Kafka consumer to obtain incremental data from the change tracking instance.	subscribe_example-master/javaimpl/src/main/java/recordgenerator/
2. Deserialize the image of the incremental data, and obtain the , , and other attributes. **Warning** • If the source instance is a self-managed Oracle database, you must enable supplemental logging for all columns. This ensures that the client can successfully consume the tracked data and guarantees the integrity of the pre-image and post-image. You can submit a ticket to enable supplemental logging for all columns. • If the source instance is not a self-managed Oracle database, DTS does not guarantee the integrity of the pre-image. We recommend that you verify the obtained pre-image. pre-imagepost-image	subscribe_example-master/javaimpl/src/main/java/boot/MysqlRecordPrinter.java

Step	File or directory
3. Convert the dataTypeNumber values in the deserialized data into data types of the corresponding database. Note For more information, see the following topics: - Mappings between MySQL data types and dataTypeNumber values - Mappings between Oracle data types and dataTypeNumber values - Mappings between PostgreSQL data types and dataTypeNumber values	subscribe_example-master/javaimpl/src/main/java/recordprocessor/mysql/

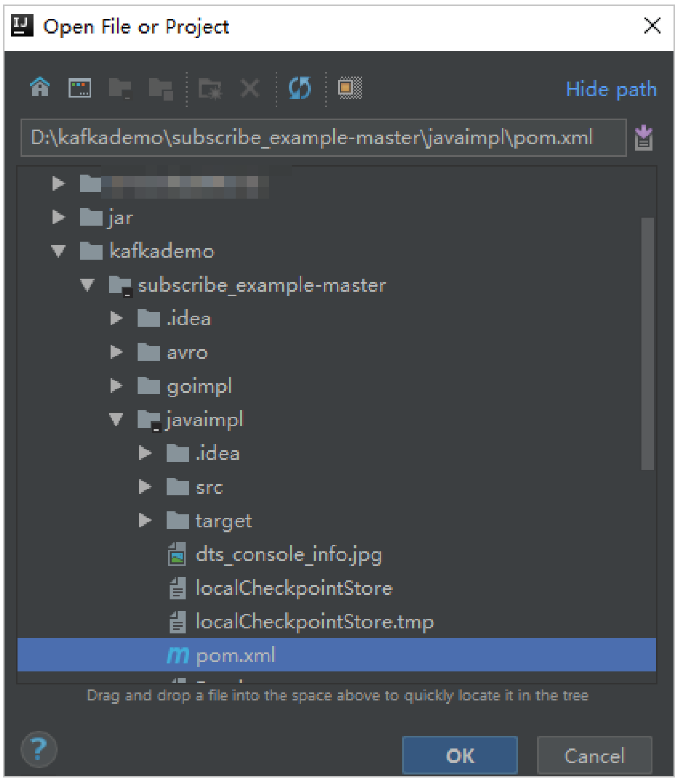
Procedure

The following steps show how to run the Kafka client to consume tracked data. In this example, IntelliJ IDEA (Community Edition 2018.1.4 Windows) is used.

1. Create a change tracking task. For more information, see Track data changes from an ApsaraDB RDS for MySQL instance, Track data changes from a PolarDB for MySQL cluster, or Track data changes from a self-managed Oracle database.
2. Create one or more consumer groups. For more information, see Create consumer groups.
3. Download the demo code of the Kafka client and decompress the package.
4. Open IntelliJ IDEA. In the window that appears, click **Open**.



5. In the dialog box that is displayed, go to the directory where the downloaded demo code resides. Find the *pom.xml* file.

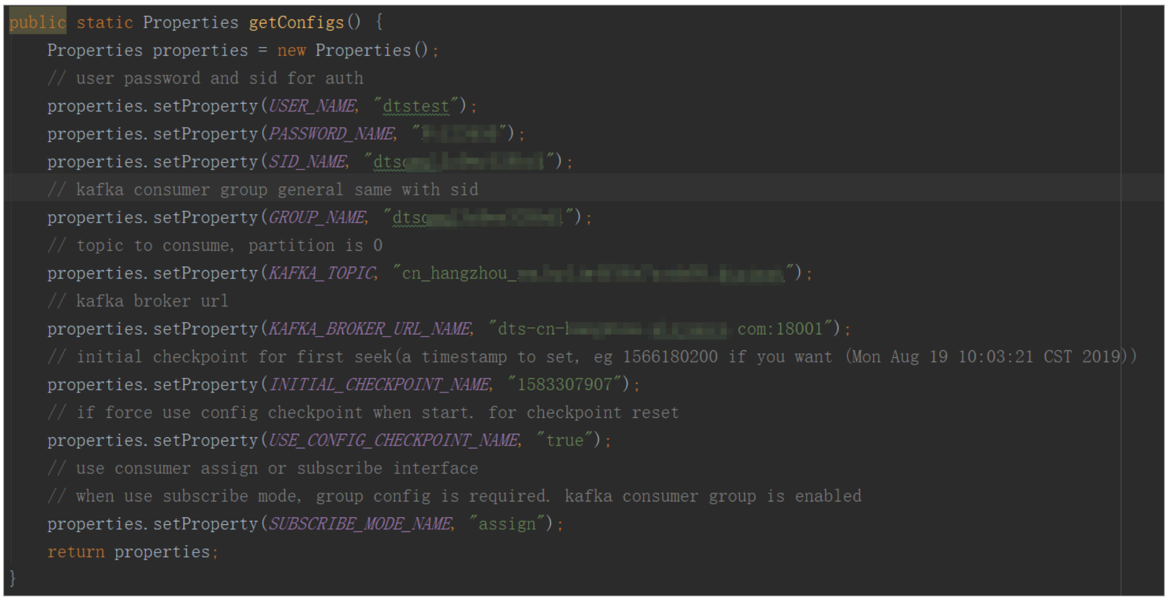


Open File or Project

D:\kafkademo\subscribe_example-master\javaimpl\pom.xml

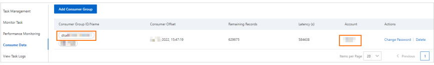
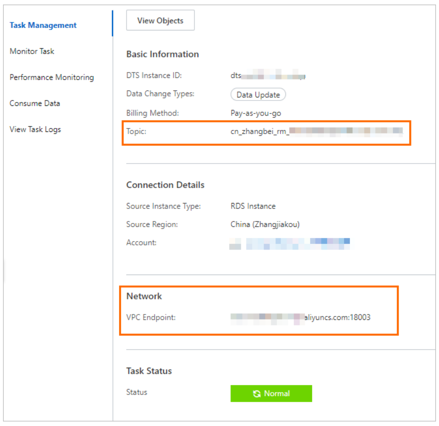
Drag and drop a file into the space above to quickly locate it in the tree

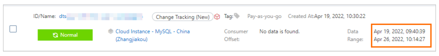
6. In the dialog box that is displayed, select **Open as Project**.
7. In the Project tool window of IntelliJ IDEA, expand folders to find the demo file of the Kafka client, and double-click the file. The file name is *NotifyDemoDB.java*.
8. Set the parameters in the *NotifyDemoDB.java* file.



```
public static Properties getConfigs() {
    Properties properties = new Properties();
    // user password and sid for auth
    properties.setProperty(USER_NAME, "dtstest");
    properties.setProperty(PASSWORD_NAME, "[redacted]");
    properties.setProperty(SID_NAME, "dts[redacted]");
    // kafka consumer group general same with sid
    properties.setProperty(GROUP_NAME, "dts[redacted]");
    // topic to consume, partition is 0
    properties.setProperty(KAFKA_TOPIC, "cn_hangzhou_[redacted]");
    // kafka broker url
    properties.setProperty(KAFKA_BROKER_URL_NAME, "dts-cn-[redacted] com:18001");
    // initial checkpoint for first seek(a timestamp to set, eg 1566180200 if you want (Mon Aug 19 10:03:21 CST 2019))
    properties.setProperty(INITIAL_CHECKPOINT_NAME, "1583307907");
    // if force use config checkpoint when start. for checkpoint reset
    properties.setProperty(USE_CONFIG_CHECKPOINT_NAME, "true");
    // use consumer assign or subscribe interface
    // when use subscribe mode, group config is required. kafka consumer group is enabled
    properties.setProperty(SUBSCRIBE_MODE_NAME, "assign");
    return properties;
}
```

Parameter	Description	Method to obtain

Parameter	Description	Method to obtain
USER_NAME	The username of the consumer group. **Warning** If you are not using the SDK client that is used in this topic, you must specify this parameter in the following format: `<Username>-<Consumer group ID>`. Example: `dtstest-dtsae******bpv`. Otherwise, the connection fails.	In the DTS console, click the instance ID. In the left-side navigation pane, click **Consume Data**. You can obtain the **Consumer Group ID** and the corresponding **Account**. **Note** The password of the consumer group account is automatically specified when you create a consumer group.  Task Management Monitor Task Performance Monitoring Consume Data View Task Logs Add Consumer Group Change Password | Delete Items per Page 20 Previous 1 Consumer Group ID/Name | Consumer Offset | Remaining Records | Latency (s) | Account | Actions dts[redacted] [redacted] | [redacted]2022, 15:47:19 | 629675 | 584438 | [redacted] | Change Password | Delete
PASSWORD_NAME	The password of the account.	
SID_NAME	The ID of the consumer group.	
GROUP_NAME	The name of the consumer group. Set this parameter to the consumer group ID.	
KAFKA_TOPIC	The topic of the change tracking instance.	In the DTS console, click the instance ID. On the **Task Management** page, you can obtain the tracked **Topic** and the network information.  Task Management Monitor Task Performance Monitoring Consume Data View Task Logs View Objects Basic Information DTS Instance ID: dts[redacted] Data Change Types: Data Update Billing Method: Pay-as-you-go Topic: cn_zhangbei_rm_[redacted] Connection Details Source Instance Type: RDS Instance Source Region: China (Zhangjiakou) Account: [redacted] Network VPC Endpoint: [redacted]aliyuncs.com:18003 Task Status Status Normal
KAFKA_BROKER_URL_NAME	The endpoint of the change tracking instance. **Note** If you track data changes over internal networks, the network latency is minimal. This is applicable if the Elastic Compute Service (ECS) instance where you deploy the Kafka client belongs to the classic network or the same virtual private cloud (VPC) as the change tracking instance.	

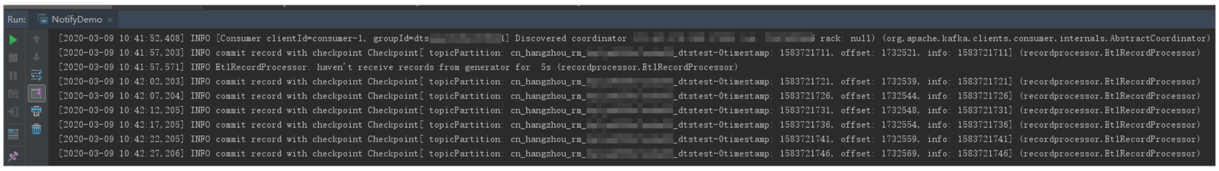
Parameter	Description	Method to obtain
INITIAL_CHECKPOINT_NAME	The consumer offset of consumed data. The value is a UNIX timestamp. Example: 1592269238. Note You must save the consumer offset. ◦ If the consumption process is interrupted, you can specify the consumer offset on the change tracking client to resume data consumption. This prevents data loss. ◦ When you start the change tracking client, you can specify the consumer offset to consume data on demand.	The consumer offset of consumed data must be within the data range of the change tracking instance, as shown in the following figure. The consumer offset must be converted into a UNIX timestamp.  ID/Name: dts[redacted] Change Tracking (New) Tag: Pay-as-you-go Created At:Apr 19, 2022, 10:30:22 □ Normal Cloud Instance - MySQL - China (Zhangjiakou) Consumer Offset: No data is found. Data Range: Apr 19, 2022, 09:40:39 Apr 26, 2022, 10:14:27 Note ◦ For more information about how to view the data range, see View tracked data changes. ◦ You can use a search engine to obtain a UNIX timestamp converter.
USE_CONFIG_CHECKPOINT_NAME	Specifies whether the client is forced to consume data from the specified consumer offset. Default value: *true*. The default value retains the data that is received but not processed.	N/A
SUBSCRIBE_MODE_NAME	Specifies whether to run two or more Kafka clients for a consumer group. If you want to use this feature, set this parameter to *subscribe* for these Kafke clients. Default value: *assign*. The default value indicates that only one Kafka client is deployed.	N/A

9. In the top menu bar of IntelliJ IDEA, choose **Run > Run** to run the client.

> Note The first time you run IntelliJ IDEA, a specific time period is required to load and install the relevant dependency.

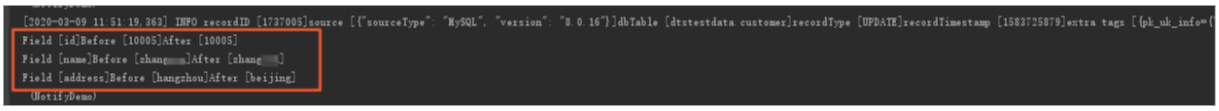
Running result of the Kafka client

The following figure shows the result that the Kafka client can track data changes from the source database.



```
[2020-03-09 10:41:52,408] INFO [Consumer clientId=consumer-1, groupId=dts[redacted]l] Discovered coordinator [redacted]6 rack: null) (org.apache.kafka.clients.consumer.internals.AbstractCoordinator)
[2020-03-09 10:41:57,203] INFO commit record with checkpoint Checkpoint[ topicPartition: cn_hangzhou_rm_[redacted]_dtstest-0timestamp: 1583721711, offset: 1732521, info: 1583721711] (recordprocessor.EtlRecordProcessor)
[2020-03-09 10:41:57,571] INFO EtlRecordProcessor: haven't receive records from generator for  5s (recordprocessor.EtlRecordProcessor)
[2020-03-09 10:42:02,203] INFO commit record with checkpoint Checkpoint[ topicPartition: cn_hangzhou_rm_[redacted]_dtstest-0timestamp: 1583721721, offset: 1732539, info: 1583721721] (recordprocessor.EtlRecordProcessor)
[2020-03-09 10:42:07,204] INFO commit record with checkpoint Checkpoint[ topicPartition: cn_hangzhou_rm_[redacted]_dtstest-0timestamp: 1583721726, offset: 1732544, info: 1583721726] (recordprocessor.EtlRecordProcessor)
[2020-03-09 10:42:12,205] INFO commit record with checkpoint Checkpoint[ topicPartition: cn_hangzhou_rm_[redacted]_dtstest-0timestamp: 1583721731, offset: 1732548, info: 1583721731] (recordprocessor.EtlRecordProcessor)
[2020-03-09 10:42:17,205] INFO commit record with checkpoint Checkpoint[ topicPartition: cn_hangzhou_rm_[redacted]_dtstest-0timestamp: 1583721736, offset: 1732554, info: 1583721736] (recordprocessor.EtlRecordProcessor)
[2020-03-09 10:42:22,205] INFO commit record with checkpoint Checkpoint[ topicPartition: cn_hangzhou_rm_[redacted]_dtstest-0timestamp: 1583721741, offset: 1732559, info: 1583721741] (recordprocessor.EtlRecordProcessor)
[2020-03-09 10:42:27,206] INFO commit record with checkpoint Checkpoint[ topicPartition: cn_hangzhou_rm_[redacted]_dtstest-0timestamp: 1583721746, offset: 1732569, info: 1583721746] (recordprocessor.EtlRecordProcessor)
```

You can delete the `//` characters from the `//log.info(ret);` string in line 25 of the *NotifyDemoDB.java* file. Then, run the client again to view the data change information.



```
[2020-03-09 11:51:19,363] INFO recordID [1737005]source [{"sourceType": "MySQL", "version": "8.0.16"}]dbTable [dtstestdata.customer]recordType [UPDATE]recordTimestamp [1583725879]extra tags [{pk_uk_info={
Field [id]Before [10005]After [10005]
Field [name]Before [zhang[redacted]]After [zhang[redacted]]
Field [address]Before [hangzhou]After [beijing]
 (NotifyDemo)
```

FAQ

- Q: Why do I need to record the consumer offset of the Kafka client?

  A: The consumer offset recorded by DTS is the time when DTS receives a commit operation from the Kafka client. The recorded consumer offset may be different from the actual consumption time. If a business application or the Kafka client is unexpectedly interrupted, you can specify an accurate consumer offset to continue data consumption. This prevents data loss or duplicate data consumption.

Mappings between MySQL data types and dataTypeNumber values

MySQL data type	Value of dataTypeNumber
MYSQL_TYPE_DECIMAL	0
MYSQL_TYPE_INT8	1
MYSQL_TYPE_INT16	2
MYSQL_TYPE_INT32	3
MYSQL_TYPE_FLOAT	4
MYSQL_TYPE_DOUBLE	5
MYSQL_TYPE_NULL	6
MYSQL_TYPE_TIMESTAMP	7
MYSQL_TYPE_INT64	8
MYSQL_TYPE_INT24	9
MYSQL_TYPE_DATE	10
MYSQL_TYPE_TIME	11
MYSQL_TYPE_DATETIME	12
MYSQL_TYPE_YEAR	13
MYSQL_TYPE_DATE_NEW	14

MySQL data type	Value of dataTypeNumber
MYSQL_TYPE_VARCHAR	15
MYSQL_TYPE_BIT	16
MYSQL_TYPE_TIMESTAMP_NEW	17
MYSQL_TYPE_DATETIME_NEW	18
MYSQL_TYPE_TIME_NEW	19
MYSQL_TYPE_JSON	245
MYSQL_TYPE_DECIMAL_NEW	246
MYSQL_TYPE_ENUM	247
MYSQL_TYPE_SET	248
MYSQL_TYPE_TINY_BLOB	249
MYSQL_TYPE_MEDIUM_BLOB	250
MYSQL_TYPE_LONG_BLOB	251
MYSQL_TYPE_BLOB	252
MYSQL_TYPE_VAR_STRING	253
MYSQL_TYPE_STRING	254
MYSQL_TYPE_GEOMETRY	255

Mappings between Oracle data types and dataTypeNumber values

Oracle data type	Value of dataTypeNumber
VARCHAR2/NVARCHAR2	1
NUMBER/FLOAT	2
LONG	8
DATE	12
RAW	23
LONG_RAW	24
UNDEFINED	29
XMLTYPE	58

Oracle data type	Value of dataTypeNumber
ROWID	69
CHAR and NCHAR	96
BINARY_FLOAT	100
BINARY_DOUBLE	101
CLOB/NCLOB	112
BLOB	113
BFILE	114
TIMESTAMP	180
TIMESTAMP_WITH_TIME_ZONE	181
INTERVAL_YEAR_TO_MONTH	182
INTERVAL_DAY_TO_SECOND	183
UROWID	208
TIMESTAMP_WITH_LOCAL_TIME_ZONE	231

Mappings between PostgreSQL data types and dataTypeNumber values

PostgreSQL data type	Value of dataTypeNumber
INT2/SMALLINT	21
INT4/INTEGER/SERIAL	23
INT8/BIGINT	20
CHARACTER	18
CHARACTER VARYING	1043
REAL	700
DOUBLE PRECISION	701
NUMERIC	1700
MONEY	790
DATE	1082

PostgreSQL data type	Value of dataTypeNumber
TIME/TIME WITHOUT TIME ZONE	1083
TIME WITH TIME ZONE	1266
TIMESTAMP/TIMESTAMP WITHOUT TIME ZONE	1114
TIMESTAMP WITH TIME ZONE	1184
BYTEA	17
TEXT	25
JSON	114
JSONB	3082
XML	142
UUID	2950
POINT	600
LSEG	601
PATH	602
BOX	603
POLYGON	604
LINE	628
CIDR	650
CIRCLE	718
MACADDR	829
INET	869
INTERVAL	1186
TXID_SNAPSHOT	2970
PG_LSN	3220
TSVECTOR	3614
TSQUERY	3615

4.Change tracking (outdated)

4.1. Track data changes from a PolarDB-X 1.0 instance

Data Transmission Service (DTS) allows you to track data changes from databases in real time. You can use the change tracking feature in the following scenarios: lightweight cache updates, business decoupling and asynchronous data processing, and synchronization of extract, transform, and load (ETL) operations. This topic describes how to track data changes from a PolarDB-X 1.0 instance (formerly DRDS).

Prerequisites

A database is created in the PolarDB-X 1.0 instance based on one or more ApsaraDB RDS for MySQL instances, and the database version number is 5.5 or 5.6.

> **Note** DTS does not support PolarDB-X 1.0 databases that are created based on PolarDB for MySQL clusters.

Precautions

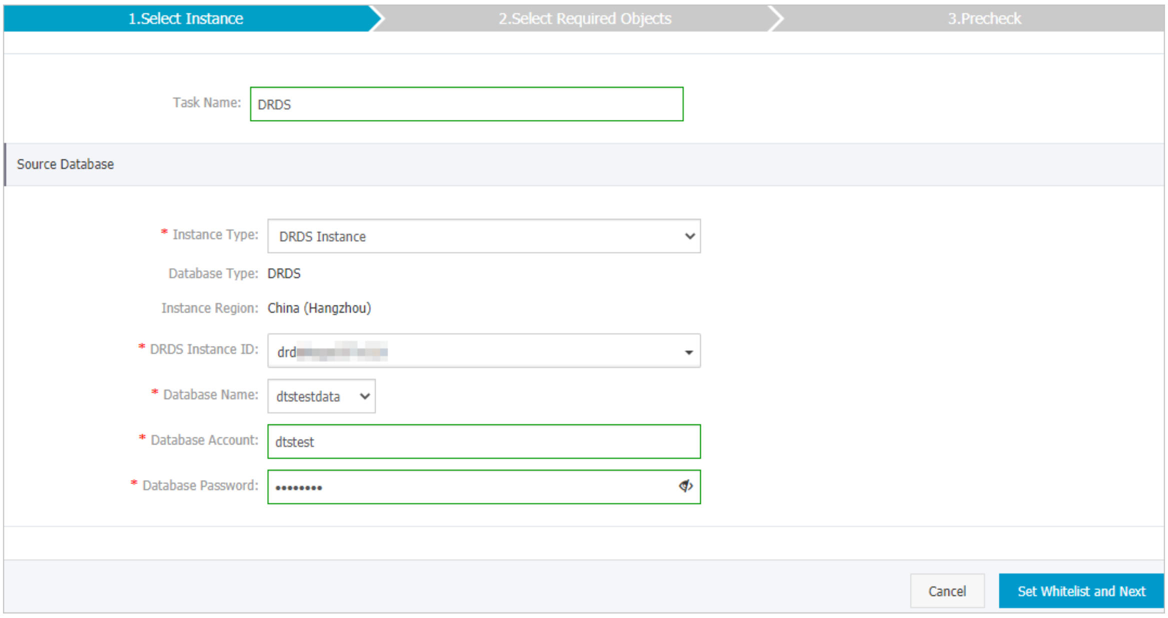
-
- When the change tracking task is running, do not scale the capacity of the PolarDB-X 1.0 instance, migrate frequently-accessed tables, change shard keys, or perform DDL operations on source objects. Otherwise, the change tracking task fails.
- After the change tracking task is configured, you cannot reselect the objects for change tracking. For more information, see Modify objects for change tracking.
- You can only track data changes of specific tables.
- You cannot track the of the source instance. schema updates

Procedure

1. Create a change tracking instance. For more information, see Purchase procedure.

   > **Note** On the buy page, set **Instance Type** to **PolarDB-X (formerly DRDS)**.

2. Log on to the DTS console.
3. In the left-side navigation pane, click **Change Tracking**.
4. In the upper part of the **Change Tracking Tasks** page, select the region where the change tracking instance resides.
5. Find the change tracking instance and click **Configure Task** in the Actions column.
6. Configure the source database for the change tracking task.



1.Select Instance

2.Select Required Objects

3.Precheck

Task Name: DRDS

Source Database

* Instance Type: DRDS Instance

Database Type: DRDS

Instance Region: China (Hangzhou)

* DRDS Instance ID: drd[redacted]

* Database Name: dtstestdata

* Database Account: dtstest

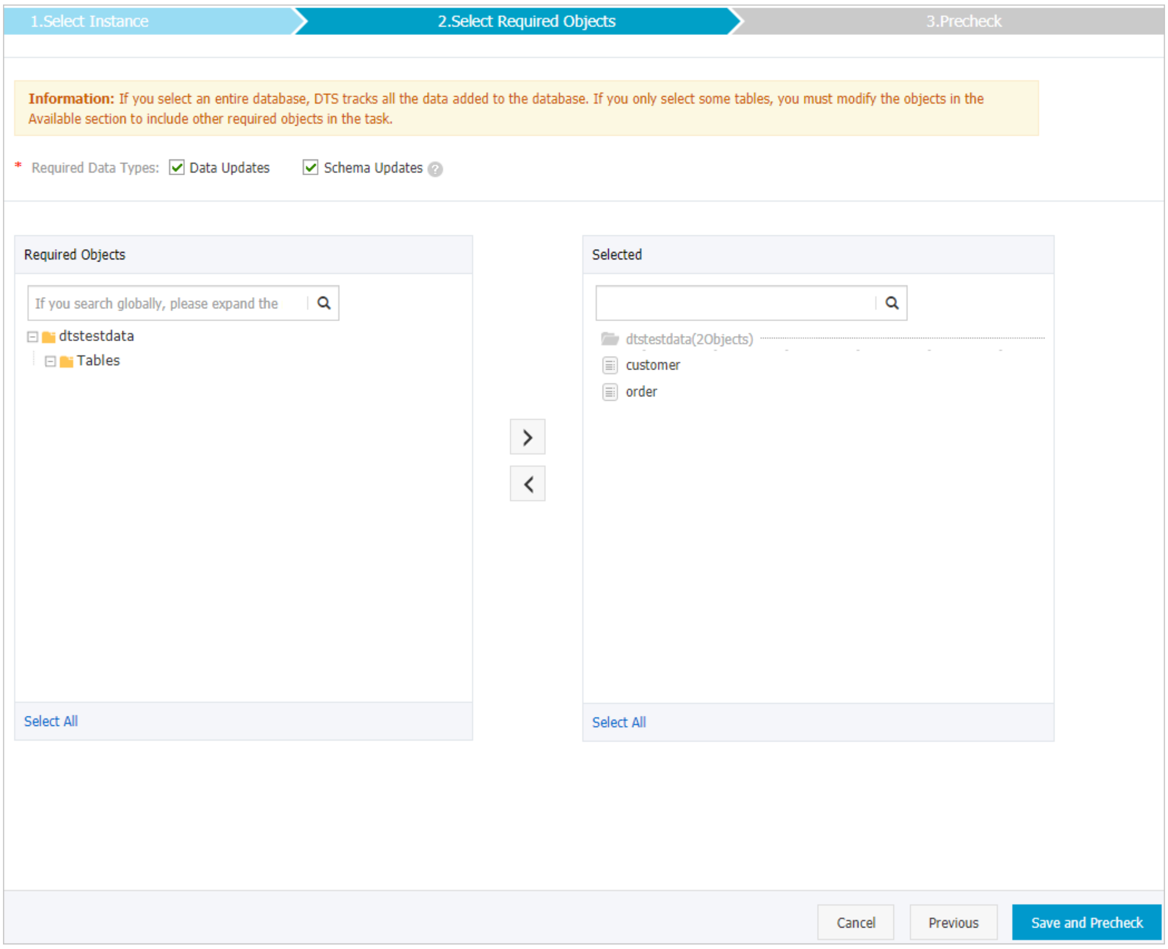
* Database Password: ••••••••

Section	Parameter	Description
N/A	Task Name	The task name that DTS automatically generates. We recommend that you specify a descriptive name that makes it easy to identify the task. You do not need to use a unique task name.
Source Database	Instance Type	Select **DRDS Instance**.
	Database Type	The value of this parameter is set to **DRDS** and cannot be changed.
	Instance Region	The **source region** that you selected on the buy page. The value of this parameter cannot be changed.
	DRDS Instance ID	The ID of the source PolarDB-X 1.0 instance.
	Database Name	The name of the database from which you want to track data changes.
	Database Account	The database account of the source PolarDB-X 1.0 instance.
	Database Password	The password of the database account.

7. In the lower-right corner of the page, click **Set Whitelist and Next**.

> **Warning**
>
> - If the source or destination database instance is an Alibaba Cloud database instance, such as an ApsaraDB RDS for MySQL or ApsaraDB for MongoDB instance, or is a self-managed database hosted on Elastic Compute Service (ECS), DTS automatically adds the CIDR blocks of DTS servers to the whitelist of the database instance or ECS security group rules. For more information, see Add the CIDR blocks of DTS servers to the security settings of on-premises databases. If the source or destination database is a self-managed database on data centers or is from other cloud service providers, you must manually add the CIDR blocks of DTS servers to allow DTS to access the database.
> - If the CIDR blocks of DTS servers are automatically or manually added to the whitelist of the database instance or ECS security group rules, security risks may arise. Therefore, before you use DTS to migrate data, you must understand and acknowledge the potential risks and take preventive measures, including but not limited to the following measures: enhance the security of your account and password, limit the ports that are exposed, authenticate API calls, regularly check the whitelist or ECS security group rules and forbid unauthorized CIDR blocks, or connect the database to DTS by using Express Connect, VPN Gateway, or Smart Access Gateway.
> - After your DTS task is completed or released, we recommend that you manually detect and remove the added CIDR blocks from the whitelist of the database instance or ECS security group rules.

8. Select the data change types and objects.



Information: If you select an entire database, DTS tracks all the data added to the database. If you only select some tables, you must modify the objects in the Available section to include other required objects in the task.

* Required Data Types: ☑ Data Updates ☑ Schema Updates

Parameter	Description
Required Data Types	The data change types that you want to track. You can select only **Data Updates**. DTS tracks data updates of the objects that you select. Data updates include the INSERT, DELETE, and UPDATE operations.
Required Objects	The objects for change tracking. Select one or more objects from the **Required Objects** section and click the > icon to add the objects to the **Selected** section. Note ○ You can select tables as the objects for change tracking. Databases cannot be selected as the objects for change tracking. ○ After the change tracking task is configured, the objects for change tracking cannot be modified. If you want to track data changes of tables that are not included in the selected objects, you must create another change tracking task for the tables.

9. In the lower-right corner of the page, click **Save and Precheck**.

> Note
>
> - Before you can start the change tracking task, DTS performs a precheck. You can start the change tracking task only after the task passes the precheck.
> - If the task fails to pass the precheck, you can click the ⓘ icon next to each failed item to view details. After you troubleshoot the issues based on the causes, you can run a precheck again.

10. Close the **Precheck** dialog box after the following message is displayed: **Precheck Passed**.

What's next

After the change tracking task is configured, DTS performs initial change tracking. After initial change tracking is complete, you can consume the tracked data. For more information, see Use the SDK demo code to consume tracked data.

4.2. Edit the consumption checkpoint

During data consumption, you can edit the consumption checkpoint of a change tracking task based on your business requirements. After you edit the consumption checkpoint, the downstream SDK client will consume the data that is generated after the specified time.

Prerequisites

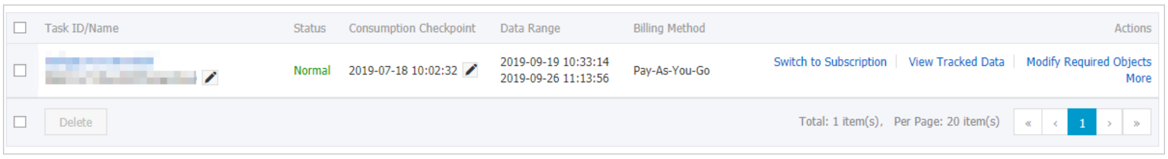
- A change tracking task is created. For more information, see Track data changes from a PolarDB-XPolarDB-X 1.0 instance.
- The change tracking task is in the **Normal** or **Error** state.

Precautions

To perform this operation, you must restart all downstream SDK consumption processes. We recommend that you perform this operation during off-peak hours to prevent service interruptions.

Procedure

1. Stop all downstream SDK consumption processes.
2.
3.
4.
5. Find the change tracking task, move the pointer over the **Consumption Checkpoint** column, and then click the ✎ icon.

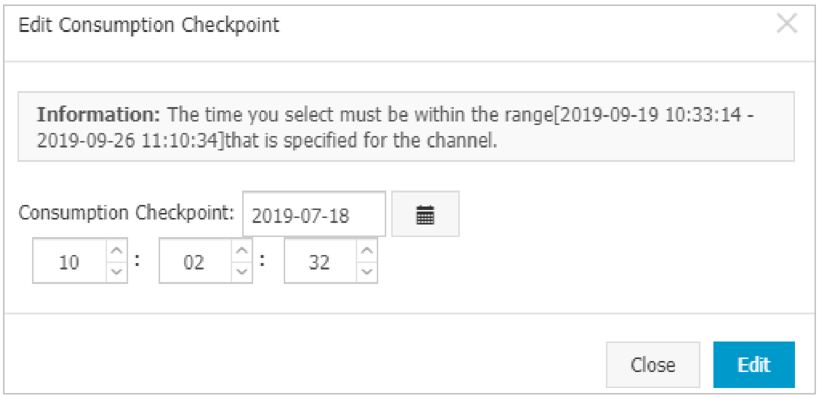


☐	Task ID/Name	Status	Consumption Checkpoint	Data Range	Billing Method	Actions
☐	[redacted] ✎	Normal	2019-07-18 10:02:32 ✎	2019-09-19 10:33:14 2019-09-26 11:13:56	Pay-As-You-Go	Switch to Subscription | View Tracked Data | Modify Required Objects More

☐ Delete — Total: 1 item(s), Per Page: 20 item(s) « ‹ 1 › »

6. In the **Edit Consumption Checkpoint** dialog box, specify a new consumption checkpoint.

> **Note** The selected time range must be within the time range of all tracked data changes. For more information, see the prompt in the dialog box.



Edit Consumption Checkpoint

Information: The time you select must be within the range[2019-09-19 10:33:14 - 2019-09-26 11:10:34]that is specified for the channel.

Consumption Checkpoint: 2019-07-18

10 : 02 : 32

7. Click **Edit**.
8. Restart the downstream SDK consumption processes.

References

Use the SDK demo code to consume tracked data

4.3. Modify objects for change tracking

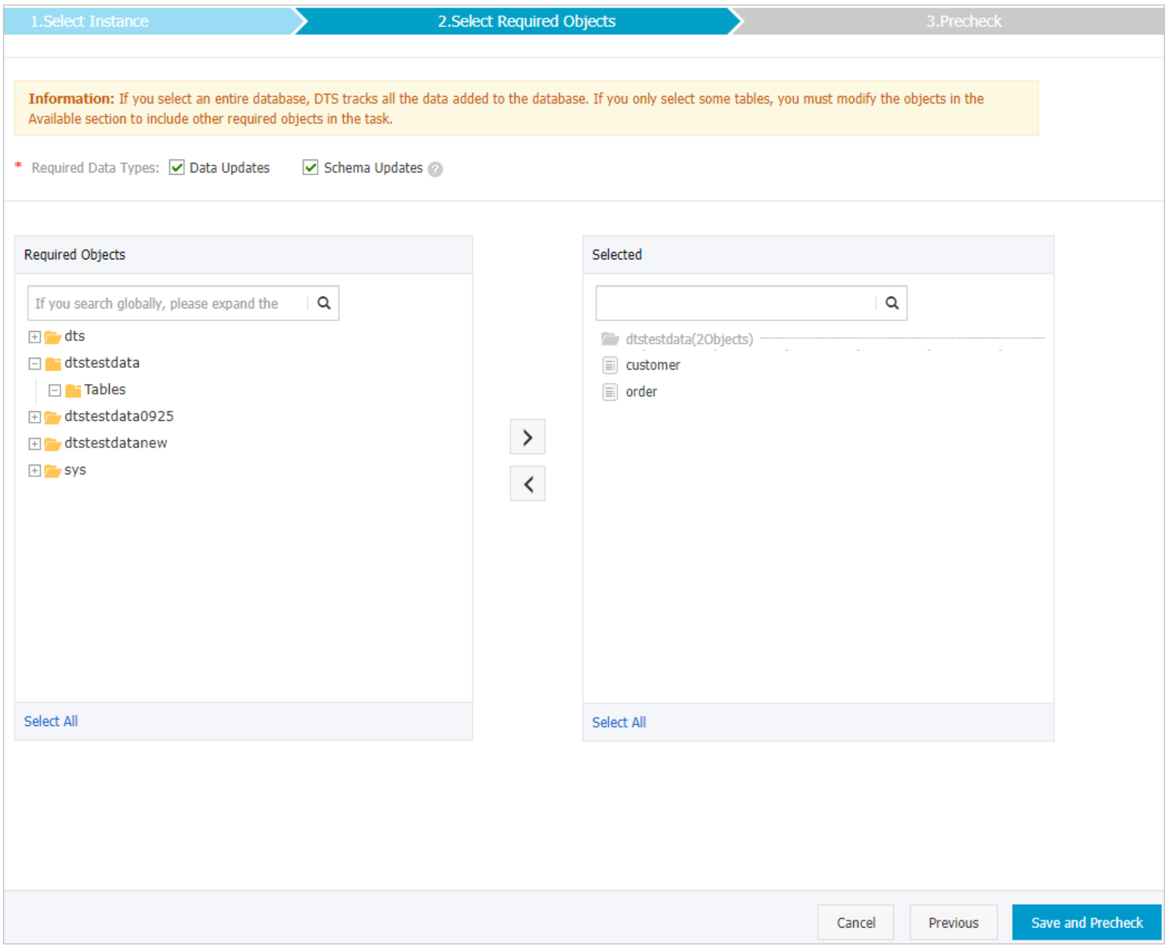
DTS allows you to add or remove objects for change tracking in the consumption process. This topic describes how to modify objects for change tracking.

Note

- After you add an object, the change tracking task pulls the incremental data of the new object from the time when the modification takes effect.
- After you remove an object, if the change tracking client can tracks data changes of the removed object, you need to filter the object in the change tracking client.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Change Tracking**.
3. In the upper part of the **Change Tracking Tasks** page, select the region where the change tracking instance resides.
4. Find the change tracking instance and click the instance ID.
5. Find the change tracking instance and click **Modify Required Objects** in the **Actions** column.
6. In the **Select Required Objects** step, you can add or remove the objects for change tracking.



Information: If you select an entire database, DTS tracks all the data added to the database. If you only select some tables, you must modify the objects in the Available section to include other required objects in the task.

* Required Data Types: ☑ Data Updates ☑ Schema Updates

   - Add the objects for change tracking

     In the **Required Objects** section, select one or more objects and click the > icon to add the objects to the **Selected** section.

   - Remove the objects for change tracking

In the **Selected** section, select one or more objects and click the < icon to move the objects to the **Required Objects** section.

7. In the lower-right corner of the page, click **Save and Precheck**.

> ⓘ Note
>
> - Before you can start the change tracking task, a precheck is performed. You can start the change tracking task only after the task passes the precheck.
> - If the task fails to pass the precheck, click the ⓘ icon next to each failed item to view details. Troubleshoot the issues based on the causes and run a precheck again.

8. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**

Subsequent operations

After modifying the objects for change tracking, you must restart the downstream SDK client to apply the modifications.

4.4. View tracked data changes

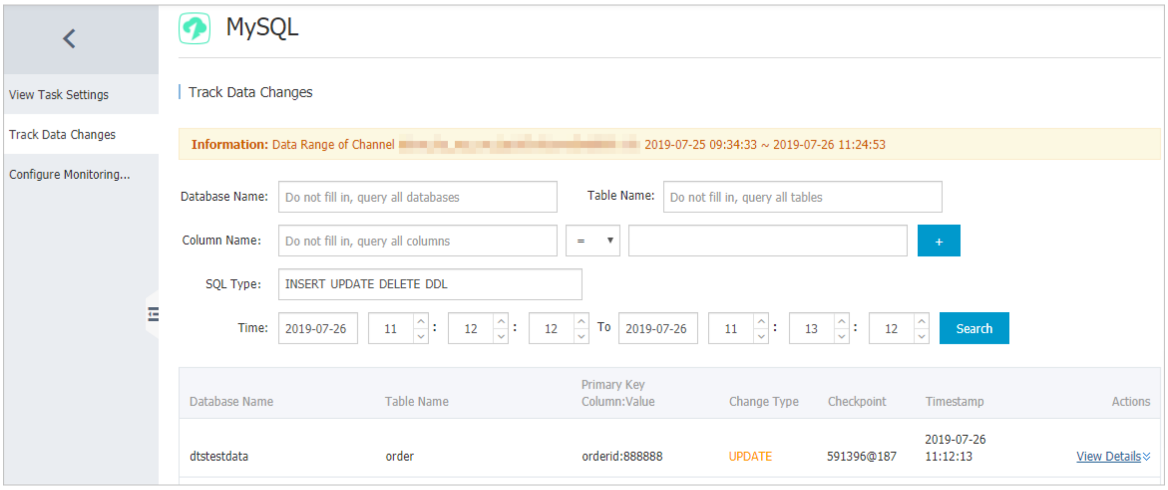
In the Data Transmission Service (DTS) console, you can view the data changes that are tracked by a change tracking task. This topic provides the definition of each field in the tracked data changes.

Prerequisites

- A change tracking task is created. For more information, see Track data changes from a PolarDB-XPolarDB-X 1.0 instance.
- The change tracking task is in the Normal or Error state.

Procedure

1.
2.
3. At the top of the **Change Tracking Tasks** page, select the region where the change tracking instance resides.
4. Find the change tracking instance and click the instance ID.
5. In the left-side navigation pane, click **Track Data Changes**.
6. On the **Track Data Changes** page, 20 data entries that are tracked in the last minute are displayed by default. You can specify conditions to filter the tracked data changes.



MySQL

View Task Settings

Track Data Changes

Configure Monitoring...

Track Data Changes

Information: Data Range of Channel [redacted] 2019-07-25 09:34:33 ~ 2019-07-26 11:24:53

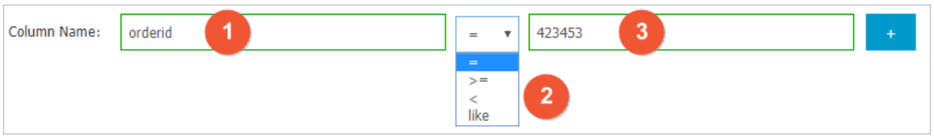
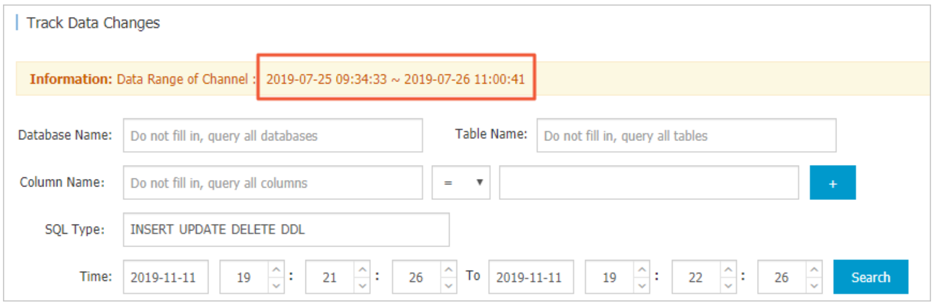
Database Name: Do not fill in, query all databases Table Name: Do not fill in, query all tables

Column Name: Do not fill in, query all columns = ▾ +

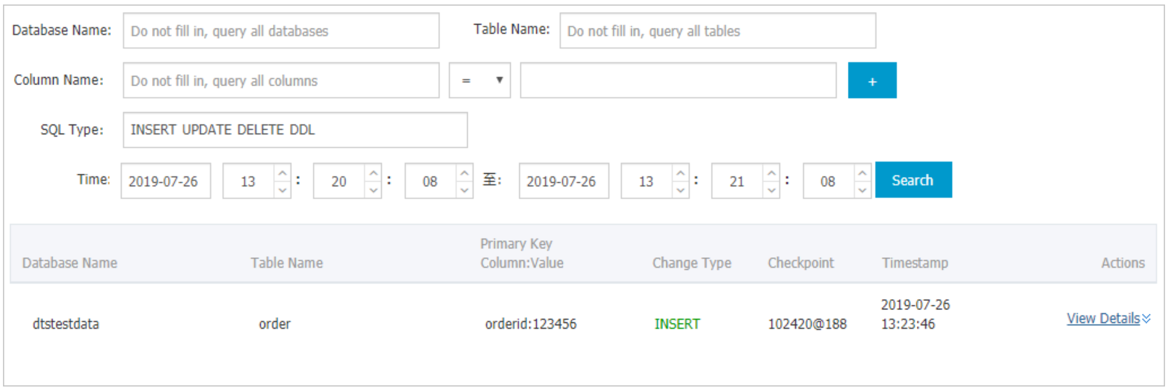
SQL Type: INSERT UPDATE DELETE DDL

Time: 2019-07-26 11 : 12 : 12 To 2019-07-26 11 : 13 : 12 Search

Database Name	Table Name	Primary Key Column:Value	Change Type	Checkpoint	Timestamp	Actions
dtstestdata	order	orderid:888888	UPDATE	591396@187	2019-07-26 11:12:13	View Details

Filter condition	Description
Database Name	Enter the name of the database that you want to query. If you do not specify this condition, all databases are queried.
Table Name	Enter the name of the table that you want to query. If you do not specify this condition, all tables are queried.
Column Name	Enter the name of the column that you want to query, select an operator, and then enter the column value. If you do not specify this condition, all columns are queried. **Note** You can click the plus sign (+) to add more column filters.  Column Name: orderid 1 = ▾ 423453 3 + = >= < like 2
SQL Type	Select the types of SQL operations that you want to query. INSERT, UPDATE, DELETE, and DDL operations are queried by default.
Time	Select a time range. The selected time range must be within the time range of all tracked data changes. For more information, see the prompt at the top of the page.  Track Data Changes **Information:** Data Range of Channel : 2019-07-25 09:34:33 ~ 2019-07-26 11:00:41 Database Name: Do not fill in, query all databases Table Name: Do not fill in, query all tables Column Name: Do not fill in, query all columns = ▾ + SQL Type: INSERT UPDATE DELETE DDL Time: 2019-11-11 19 : 21 : 26 To 2019-11-11 19 : 22 : 26 Search

7. Click **Search** to retrieve specific data entries.



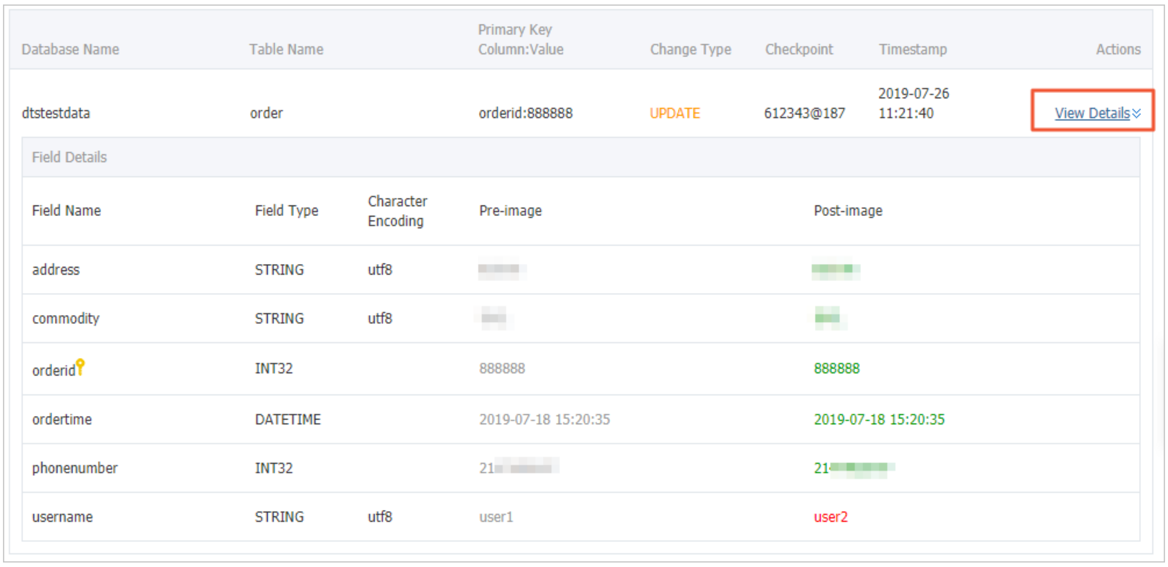
Database Name:	Do not fill in, query all databases	Table Name:	Do not fill in, query all tables
Column Name:	Do not fill in, query all columns	= ▾	+
SQL Type:	INSERT UPDATE DELETE DDL		
Time:	2019-07-26 13 : 20 : 08	至:	2019-07-26 13 : 21 : 08 Search

Database Name	Table Name	Primary Key Column:Value	Change Type	Checkpoint	Timestamp	Actions
dtstestdata	order	orderid:123456	INSERT	102420@188	2019-07-26 13:23:46	View Details

Consumption checkpoint and timestamp

Database type	Consumption checkpoint	Timestamp
ApsaraDB RDS for MySQL	The location and time of an incremental data entry in the binary log.	The timestamp when an incremental data entry is written to the binary log file.
Oracle	The location and time of an incremental data entry in the redo or archive log.	The timestamp when an incremental data entry is written to the redo or archive log.

8. Click **View Details** in the Actions column of a data entry to view the details.



Database Name	Table Name	Primary Key Column:Value	Change Type	Checkpoint	Timestamp	Actions
dtstestdata	order	orderid:888888	UPDATE	612343@187	2019-07-26 11:21:40	View Details

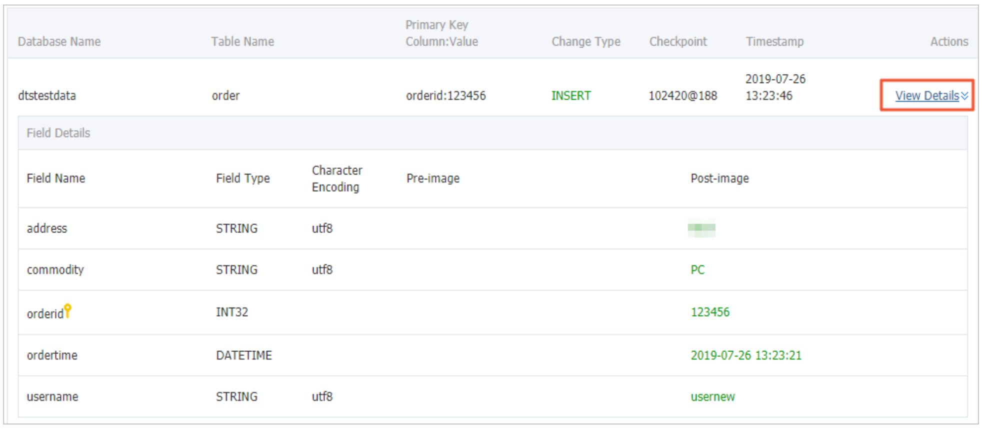
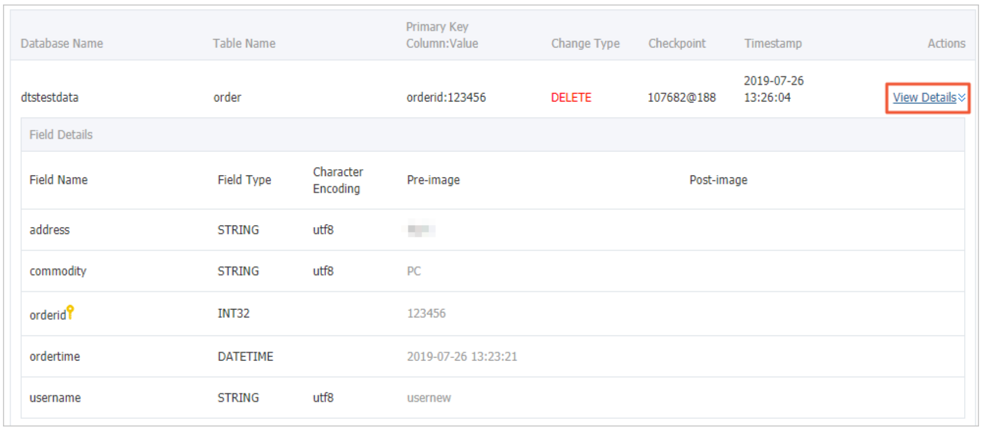
Field Details

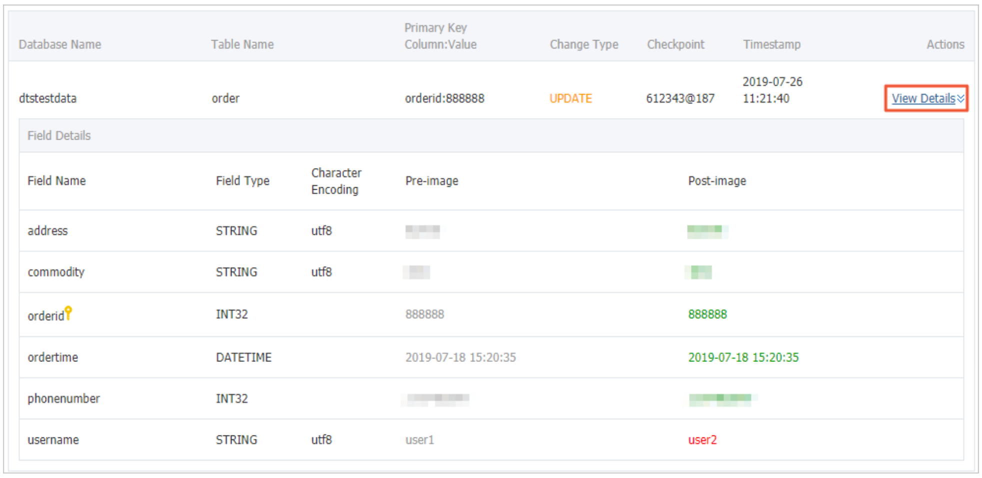
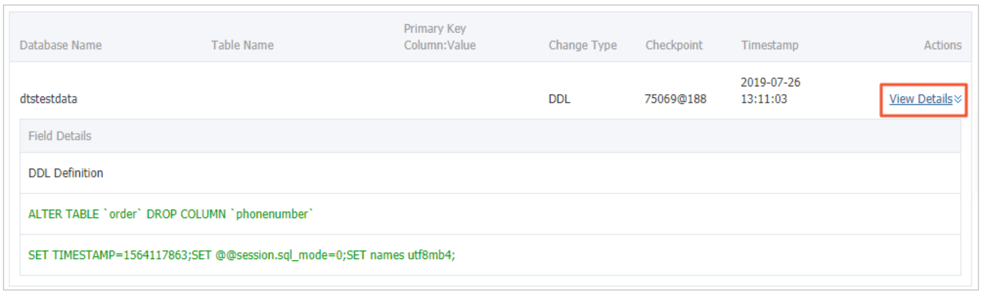
Field Name	Field Type	Character Encoding	Pre-image	Post-image
address	STRING	utf8	[redacted]	[redacted]
commodity	STRING	utf8	[redacted]	[redacted]
orderid	INT32		888888	888888
ordertime	DATETIME		2019-07-18 15:20:35	2019-07-18 15:20:35
phonenumber	INT32		21[redacted]	21[redacted]
username	STRING	utf8	user1	user2

Field	Description
Field Name	The name of the field or column.
Field Type	The type of the field.
Character Encoding	The character encoding of the field, such as UTF-8, GBK, Latin 1, or UTF-8 (MB4).

Field	Description
Pre-image	The value of each field before the data entry is updated.
Post-image	The value of each field after the data entry is updated. The updated values are displayed in red.

Additional information about the pre-image and post-image

<table>
<tr><th>Operation type</th><th>Value description</th></tr>
<tr><td>INSERT</td><td>

<table>
<tr><th>Database Name</th><th>Table Name</th><th>Primary Key Column:Value</th><th>Change Type</th><th>Checkpoint</th><th>Timestamp</th><th>Actions</th></tr>
<tr><td>dtstestdata</td><td>order</td><td>orderid:123456</td><td>INSERT</td><td>102420@188</td><td>2019-07-26 13:23:46</td><td>View Details</td></tr>
</table>
Field Details
<table>
<tr><th>Field Name</th><th>Field Type</th><th>Character Encoding</th><th>Pre-image</th><th>Post-image</th></tr>
<tr><td>address</td><td>STRING</td><td>utf8</td><td></td><td>[redacted]</td></tr>
<tr><td>commodity</td><td>STRING</td><td>utf8</td><td></td><td>PC</td></tr>
<tr><td>orderid</td><td>INT32</td><td></td><td></td><td>123456</td></tr>
<tr><td>ordertime</td><td>DATETIME</td><td></td><td></td><td>2019-07-26 13:23:21</td></tr>
<tr><td>username</td><td>STRING</td><td>utf8</td><td></td><td>usernew</td></tr>
</table>
The value of the pre-image is empty.
The value of the post-image is the data written by the INSERT operation.
</td></tr>
<tr><td>DELETE</td><td>

<table>
<tr><th>Database Name</th><th>Table Name</th><th>Primary Key Column:Value</th><th>Change Type</th><th>Checkpoint</th><th>Timestamp</th><th>Actions</th></tr>
<tr><td>dtstestdata</td><td>order</td><td>orderid:123456</td><td>DELETE</td><td>107682@188</td><td>2019-07-26 13:26:04</td><td>View Details</td></tr>
</table>
Field Details
<table>
<tr><th>Field Name</th><th>Field Type</th><th>Character Encoding</th><th>Pre-image</th><th>Post-image</th></tr>
<tr><td>address</td><td>STRING</td><td>utf8</td><td>[redacted]</td><td></td></tr>
<tr><td>commodity</td><td>STRING</td><td>utf8</td><td>PC</td><td></td></tr>
<tr><td>orderid</td><td>INT32</td><td></td><td>123456</td><td></td></tr>
<tr><td>ordertime</td><td>DATETIME</td><td></td><td>2019-07-26 13:23:21</td><td></td></tr>
<tr><td>username</td><td>STRING</td><td>utf8</td><td>usernew</td><td></td></tr>
</table>
The value of the pre-image is the data deleted by the DELETE operation.
The value of the post-image is empty.
</td></tr>
</table>

<table>
<tr><th>Operation type</th><th>Value description</th></tr>
<tr><td>UPDATE</td><td>

<table>
<tr><th>Database Name</th><th>Table Name</th><th>Primary Key Column:Value</th><th>Change Type</th><th>Checkpoint</th><th>Timestamp</th><th>Actions</th></tr>
<tr><td>dtstestdata</td><td>order</td><td>orderid:888888</td><td>UPDATE</td><td>612343@187</td><td>2019-07-26 11:21:40</td><td>View Details</td></tr>
</table>
Field Details
<table>
<tr><th>Field Name</th><th>Field Type</th><th>Character Encoding</th><th>Pre-image</th><th>Post-image</th></tr>
<tr><td>address</td><td>STRING</td><td>utf8</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>commodity</td><td>STRING</td><td>utf8</td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>orderid</td><td>INT32</td><td></td><td>888888</td><td>888888</td></tr>
<tr><td>ordertime</td><td>DATETIME</td><td></td><td>2019-07-18 15:20:35</td><td>2019-07-18 15:20:35</td></tr>
<tr><td>phonenumber</td><td>INT32</td><td></td><td>[redacted]</td><td>[redacted]</td></tr>
<tr><td>username</td><td>STRING</td><td>utf8</td><td>user1</td><td>user2</td></tr>
</table>
The value of the pre-image is the data before update.
The value of the post-image is the data after update.
Note The updated values are displayed in red.
</td></tr>
<tr><td>DDL or DML</td><td>

<table>
<tr><th>Database Name</th><th>Table Name</th><th>Primary Key Column:Value</th><th>Change Type</th><th>Checkpoint</th><th>Timestamp</th><th>Actions</th></tr>
<tr><td>dtstestdata</td><td></td><td></td><td>DDL</td><td>75069@188</td><td>2019-07-26 13:11:03</td><td>View Details</td></tr>
</table>
Field Details
DDL Definition
ALTER TABLE `order` DROP COLUMN `phonenumber`
SET TIMESTAMP=1564117863;SET @@session.sql_mode=0;SET names utf8mb4;
The details contain only DDL or DML statements.
</td></tr>
</table>

4.5. Methods of the SDK demo code

You can use the SDK demo code that is provided by DTS to consume tracked data. This topic describes the methods that are available for the SDK classes.

Download the SDK demo code

For more information, visit Download SDK.

Methods of the RegionContex class

Method	Description
`setAccessKey(accessKey)`	Specifies the AccessKey ID of the Alibaba Cloud account to which the source instance belongs.

Method	Description
`setSecret(AccessKeySecret)`	Specifies the AccessKey secret of the Alibaba Cloud account to which the source instance belongs.
`setUsePublicIp(usePublicIp)`	Specifies whether to track data changes over the Internet. **Note** DTS can track data changes only over the Internet. Therefore, set the usePublicIp parameter to `true`.
`context.setUseBinary(boolean useBinary)`	Specifies whether to enable the binary packaging feature. Valid values: True and False. We recommend that you enable this feature to improve consumption performance.
`context.setUseDrcNet(boolean useDrcNet)`	Specifies whether to enable the network optimization feature. Valid values: True and False. We recommend that you enable this feature to improve consumption performance.

Methods of the ClusterClient class

Method	Description
`void addConcurrentListener(ClusterListener arg0)`	Adds a downstream listener to retrieve data changes from a change tracking channel. **Note** The `ClusterListener arg0` parameter specifies an object of the `ClusterListener` class.
`void askForGUID(String arg0)`	Retrieves data changes from a change tracking channel. Set the String arg0 parameter to the ID of the change tracking instance.
`List<ClusterListener> getConcurrentListeners()`	Queries the list of listeners in a ClusterClient object. The return type is `List <ClusterListener>`.
`void start()`	Starts the SDK client to start change tracking.
`void stop()`	Stops the SDK client to stop change tracking. **Note** Data pulling and notification callback are performed in the same thread of the SDK client. If the consumption code of the notify() method contains a function that prevents signal interruptions, the stop() function may fail to terminate the SDK client.

Method of the ClusterListener class

The `void notify(List<ClusterMessage> arg0)` method specifies the consumption mode of tracked data. When the DTS SDK receives the data, it uses the notify() function to notify a ClusterListener object to consume the data. Then, the SDK displays the data on the screen.

Methods of the ClusterMessage class

> **Note** Each ClusterMessage object stores the data record of a transaction. Each data record in the transaction is stored by using a Record object.

Method	Description
`Record getRecord()`	Retrieves a change record from a ClusterMessage object. The change record contains an entry in the binary log file, such as a BEGIN, COMMIT, UPDATE, or INSERT operation.
`void ackAsConsumed`	After the data consumption is complete, you must call this method to send an ACK packet to instruct the DTS server to update the consumer offset. This ensures the integrity of the consumed data after an abnormal SDK client restarts. **Note** If a downstream SDK client restarts after a breakdown, the client resumes change tracking from the last consumer offset.

Methods of the Record class

The `String getAttribute(String key)` method retrieves the attribute values in a Record object. The following table describes the parameters that are available when you call this method.

Parameter	Description
`record_id`	The ID of the record. **Note** The record ID may not increment during the change tracking process.
`instance`	The endpoint that is used to connect to the database instance. The format is <IP address>:<Port number>.
`source_type`	The engine type of the database instance. The value is set to MySQL.
`source_category`	The type of the record. The value is set to full_recorded.
`timestamp`	The binlog timestamp that is generated when the SQL statement is executed in the source database.

Parameter	Description
`checkpoint`	The checkpoint of the binary log file. The format is `binlog_offset@binlog_file` . Note The `binlog_offset` parameter indicates the offset of a record in the binary log file. The `binlog_file` parameter indicates the numerical suffix of the binary log file. For example, if the name of a binary log file is mysql-bin.0008, the value of the binlog_file parameter is 8.
`record_type`	The operation type. Valid values: insert, update, delete, replace, ddl, begin, commit, and heartbeat. Note A heartbeat record indicates the heartbeat table that is defined by DTS. The system generates one heartbeat record per second to detect whether the change tracking channel is running as expected.
`db`	The name of the database.
`table_name`	The name of the table.
`record_recording`	The encoding format.
`primary`	The name of the primary key column. If the primary key is a composite key, separate column names with commas (,).
`fields_enc`	The encoding of each field value. Separate fields with commas (,). Note If a field value is not of the character type, the encoding of this field value is null.

The following table lists the methods that are preset in the SDK demo code. You can call these methods to retrieve the attribute values in a Record object.

Method	Description
`Type getOpt()`	Queries the operation type.
`String getCheckpoint()`	Queries the checkpoint of the binary log file.
`String gettimestamp()`	Queries the timestamp of the binary log file.
`String getDbname()`	Queries the database name.
`String getTablename()`	Queries the table name.

Method	Description
`String getPrimaryKeys()`	Queries the name of the primary key column.
`DBType getDbType()`	Queries the database type.
`String getServerId()`	Queries the endpoint that is used to connect to the database instance.
`int getFieldCount()`	Queries the number of fields.
`List<Field> getFieldList()`	Queries the definitions of all fields, the pre-change image values, and the post-change image values. For more information, see Methods of the Field class.
`Boolean isFirstInLogevent()`	Checks whether the record is the first transaction log in a large volume of data changes. The return value is True or False.

Methods of the Field class

Method	Description
`String getEncoding()`	Queries the encoding format of the field value.
`String getFieldname()`	Queries the name of the field.
`Type getType()`	Queries the data type of the field.
`ByteString getValue()`	Queries the value of the field. The return type is ByteString. If the field is not specified, the method returns `NULL`.
`Boolean isPrimary()`	Checks whether the field is a primary key column. The return value is True or False.

4.6. Run the SDK demo code

This topic describes how to run the SDK demo code that is provided by DTS.

Download the SDK demo code

Download and decompress the DtsSubscribeDemo package, use a text editor to open the *pom.xml* file, and then change the SDK version to the latest.

> **Note** You can obtain the latest version of the change tracking SDK from the Maven website. For more information, visit the Maven page of the change tracking SDK.

Initialize a RegionContext object

A `RegionContext` object stores the settings of authentication credentials and network access mode. The following code shows how to initialize a `RegionContext` object.

```
import java.util.List;
import com.aliyun.drc.clusterclient.RegionContext;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class MainClass
{
   public static void main(String[] args) throws Exception {
      // Create a RegionContext object.
      RegionContext context = new RegionContext();
      // Specify the AccessKey ID and AccessKey secret of the Alibaba Cloud account.
      context.setAccessKey("<AccessKey>");
      context.setSecret("<AccessKeySecret>");
       // Specify whether to track data changes over the Internet.
      context.setUsePublicIp(true);
      // Specify whether to transfer data by using the binary format.
      context.setUseBinary(true);
      // Enable the network optimization feature.
      context.setDrcNet(true);
        ...
    }
}
```

Initialize a ClusterClient object

A `ClusterClient` object connects to a change tracking channel and receives incremental data. The following code shows how to initialize a `ClusterClient` object.

```
import java.util.List;
import com.aliyun.drc.clusterclient.ClusterClient;
import com.aliyun.drc.clusterclient.DefaultClusterClient;
import com.aliyun.drc.clusterclient.RegionContext;
public class MainClass
{
      public static void main(String[] args) throws Exception {
        // Create a RegionContext object.
          RegionContext context = new RegionContext();
          context.setAccessKey("<AccessKey>");
          context.setSecret("<AccessKeySecret>");
          context.setUsePublicIp(true);
          // Create a ClusterClient object.
          final ClusterClient client = new DefaultClusterClient(context);
         ...
    }
}
```

Initialize a Listener object

A `Listener` object uses the `notify()` method to receive and consume the tracked data. The following code shows how to display the tracked data on the screen.

```
import com.aliyun.drc.clusterclient.ClusterClient;
import com.aliyun.drc.clusterclient.ClusterListener;
import com.aliyun.drc.clusterclient.DefaultClusterClient;
import com.aliyun.drc.clusterclient.RegionContext;
import com.aliyun.drc.clusterclient.message.ClusterMessage;
public class MainClass
{
    public static void main(String[] args) throws Exception {
        // Initialize the RegionContext object.
        ...
        // Initialize the ClusterClient object.
        ...
        ClusterListener listener = new ClusterListener(){
             @Override
             public void notify(List<ClusterMessage> messages) throws Exception {
                  for (ClusterMessage message : messages) {
                    // Display the tracked data on the screen.
                      System.out.println(message.getRecord() + ":" + message.getRecord().ge
tTablename() + ":"
                      + message.getRecord().getOpt());
                      // Call the following method to send an ACK packet to the DTS server.
                      message.ackAsConsumed();
              }
      }
     }
}
```

Note The `ackAsConsumed()` method sends the checkpoint and timestamp of the latest data record that was consumed by the DTS SDK to the DTS server. If the SDK restarts due to an error, the SDK obtains the consumption checkpoint from the DTS server. The SDK resumes data consumption from the checkpoint. This ensures that the SDK does not consume duplicate data.

Start the ClusterClient object

```
import java.util.List;
import com.aliyun.drc.clusterclient.ClusterClient;
import com.aliyun.drc.clusterclient.ClusterListener;
import com.aliyun.drc.clusterclient.DefaultClusterClient;
import com.aliyun.drc.clusterclient.RegionContext;
import com.aliyun.drc.clusterclient.message.ClusterMessage;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class MainClass
{
   public static void main(String[] args) throws Exception {
    // Initialize the RegionContext object.
    ...
    // Initialize the ClusterClient object.
    ...
    // Initialize the ClusterListener object.
    ...
    // Add a Listener class.
      client.addConcurrentListener(listener);
      // Specify the ID of the change tracking channel.
      client.askForGUID("dts_rdsr******_DSF");
      // Start a background thread. The main thread cannot exit.
      client.start();
}
```

Before you start a `ClusterClient` object, add a `Listener` class to the `ClusterClient` object. When the `ClusterClient` object pulls incremental data from the change tracking channel, it also calls the `notify()` method of the `Listener` class to consume data.

References

Use the SDK demo code to consume tracked data

4.7. Parse tracked SQL statements

You can use the DTS SDK to track data changes. DTS records the tracked data changes in a custom format. This topic describes how to parse various types of SQL statements.

Parse a DDL statement

If a data definition language (DDL) operation is performed in the source database, the operation type of the data record is DDL. The DDL statement is stored in the value of the first column. You can use the following sample code to parse the DDL statement:

```
String ddl_string;
Record.Type type=record.getOpt();
if(type.equals(Record.Type.DDL)){
    List<DataMessage.Record.Field> fields = record.getFieldList();
    ddl_string = fields.get(0).getValue().toString();
}
```

Parse an INSERT statement

If an INSERT operation is performed in the source database, the operation type of the data record is INSERT. You can use the following sample code to parse the INSERT statement:

```
StringBuilder insert_string=new StringBuilder();
Record.Type type=record.getOpt();
DataMessage.Record.Field field;
StringBuilder FieldName=new StringBuilder();
StringBuilder FieldValue = new StringBuilder();
if(type.equals(Record.Type.INSERT)){
       int i=0;
       List<DataMessage.Record.Field> fields = record.getFieldList();
       for (; i < fields.size(); i++) {
              field = fields.get(i);                                   FieldName.append('`'
+field.getFieldname().toLowerCase()+'`');
              FieldValue.append("'"+field.getValue()+"'");
              if (i ! = fields.size() - 1) {
                      FieldName.append(',');
                      FieldValue.append(',');
              }
        }
        insert_string.append("insert "+ record.getTablename()+"("+FieldName.toString()+") v
alues("+FieldValue.toString()+");");
}
```

Parse an UPDATE statement

If an UPDATE operation is performed in the source database, the operation type of the data record is UPDATE. The field values prior to the UPDATE operation are stored in `Record.getFieldList()` entries with even indexes. The field values after the UPDATE operation are stored in Record.getFieldList() entries with odd indexes.

If the UPDATE operation is performed on a table that has a primary key, you can use the following sample code to parse the UPDATE statement:

```
StringBuilder update_string=new StringBuilder();
Record.Type type=record.getOpt();
DataMessage.Record.Field field;
StringBuilder SetValue = new StringBuilder();
StringBuilder WhereCondition = new StringBuilder();
String ConditionStr;
boolean hasPk=false;
boolean pkMode=false;
boolean hasSet=false;
if(type.equals(Record.Type.UPDATE)){
    int i=0;
    DataMessage.Record.Field OldField = null;
    DataMessage.Record.Field NewField = null;
    List<DataMessage.Record.Field> fields = record.getFieldList();
    for (; i <fields.size() ; i++) {
        if (i % 2 == 0) {
            OldField = fields.get(i);
            continue;
        }
    NewField = fields.get(i);
    field = NewField;
    if (field.isPrimary()) {
        if (hasPk) {
            WhereCondition.append(" and ");
        }
        //where old value
        ConditionStr = getFieldValue(OldField);
        if(ConditionStr==null){                                         WhereCondition.appe
nd("`"+field.getFieldname().toLowerCase()+"`" + " " + "is null");
        }else{
               WhereCondition.append("`"+field.getFieldname().toLowerCase()+"`"+" = "+ "'"+
OldField.getValue()+"'");
         }
        hasPk = true;
    }
    if (hasSet) {
        SetValue.append(",");
    }
      SetValue.append("`"+field.getFieldname().toLowerCase()+"`" + " = " + "'"+field.getVal
ue()+"'");
    String setStr = getFieldValue(field);
    hasSet = true;
    }
    update_string.append("Update "+record.getTablename() +" Set " + SetValue + " Where "+Wh
ereCondition +";");
}
protected String getFieldValue(Field field) throws Exception {
    ByteString byteString = field.getValue();
    if (byteString == null) {
        return null;
    }
    else {
        String value;
        if (field.getType() == com.aliyun.drc.client.message.DataMessage.Record.Field.Type.
STRING && field.getEncoding() ! = null && field.getEncoding() ! = "ASCII") {
            value = field.getValue().toString(field.getEncoding());
        }
        else {
          value = byteString.toString();
        }
        return value;
    }
}
```

Parse a DELETE statement

If a DELETE operation is performed in the source database, the operation type of the data record is DELETE. If the DELETE operation is performed on a table that has a primary key, you can use the following sample code to parse the DELETE statement:

```
StringBuilder delete_string=new StringBuilder();
Record.Type type=record.getOpt();
DataMessage.Record.Field field;
StringBuilder FieldName=new StringBuilder();
StringBuilder FieldValue = new StringBuilder();
StringBuilder DeleteCondition = new StringBuilder();
boolean hasPk=false;
boolean pkMode=false;
if(type.equals(Record.Type.DELETE)){
   int i=0;
   List<DataMessage.Record.Field> fields = record.getFieldList();
   delete_string.append("Delete From" + record.getTablename() + "where");

   // Check whether the table has a primary key.
   if (record.getPrimaryKeys() ! = null) {
             pkMode = record.getPrimaryKeys().length() > 0 ? true : false;
   }
   for (; i < fields.size(); i++) {
            if ((pkMode && ! field.isPrimary())) {
                    continue;
            }
            if (hasPk) {
                    delete_string.append(" and ");
            }
            delete_string.append(field.getFieldname() + "=" + field.getValue());
            hasPk = true;
    }
    delete_string.append(";");
}
```

Parse a REPLACE statement

If a REPLACE operation is performed in the source database, the operation type of the data record is UPDATE or INSERT.

- If the value specified in the REPLACE statement does not exist, the operation type of the data record is INSERT.
- If the value specified in the REPLACE statement exists, the operation type of the data record is UPDATE.

Parse a BEGIN statement

If a BEGIN operation is performed in the source database, the operation type of the data record is BEGIN. You do not need to perform operations on fields because the BEGIN statement does not modify fields. You only need to check that the operation is a BEGIN operation. You can use the following sample code to parse the BEGIN statement:

```
StringBuilder sql_string = new StringBuilder();
Record.Type type = record.getOpt();
if(type.equals(Record.Type.BEGIN)){
        sql_string.append("Begin");
}
```

Parse a COMMIT statement

If a COMMIT operation is performed in the source database, the operation type for the data record is COMMIT. You do not need to perform operations on fields because the COMMIT statement does not modify fields. You only need to check that the operation is a COMMIT operation. You can use the following sample code to parse the COMMIT statement:

```
StringBuilder sql_string = new StringBuilder();
Record.Type type = record.getOpt();
if(type.equals(Record.Type.COMMIT)){
        sql_string.append("commit");
}
```

4.8. Use the SDK demo code to consume tracked data

After you configure a change tracking task, you can use the SDK demo code that is provided by Data Transmission Service (DTS) to track and consume data. This topic describes how to use the SDK demo code to consume tracked data.

Prerequisites

A change tracking task (the previous version) is configured. For more information, see Track data changes from a PolarDB-XPolarDB-X 1.0 instance.

Before you begin

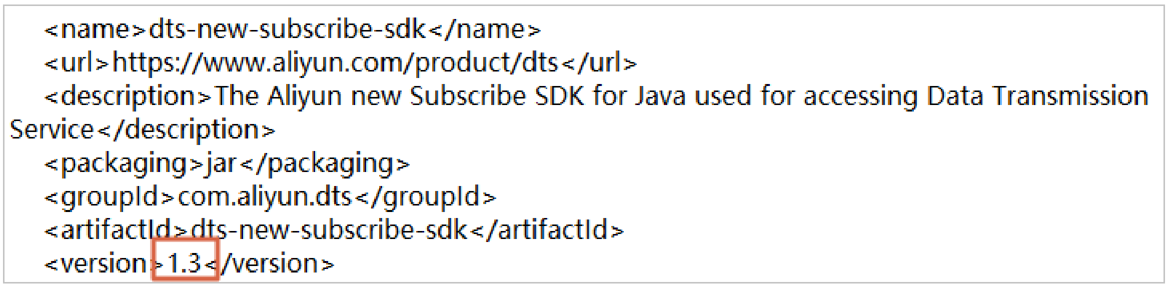
Create an AccessKey pair, which consists of an AccessKey ID and AccessKey secret. For more information, see Create an AccessKey pair.

Notice If you track and consume data as a RAM user, the **AliyunDTSFullAccess** permission must be granted to the RAM user. For more information, see Use a system policy to authorize a RAM user to manage DTS instances.

Procedure

IntelliJ IDEA (Community Edition 2018.1.4 Windows) is used in this example.

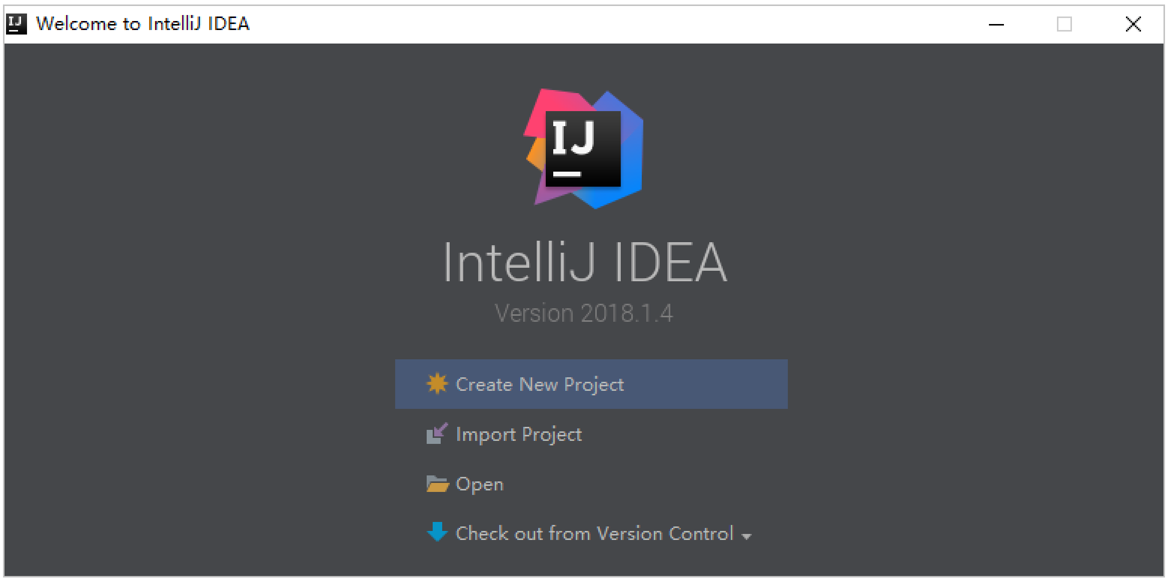
1. Download the SDK demo code package and decompress the package.
2. (Optional)Go to the directory where the package is decompressed, open the *pom.xml* file by using a text editor, and then change the SDK version to the latest.



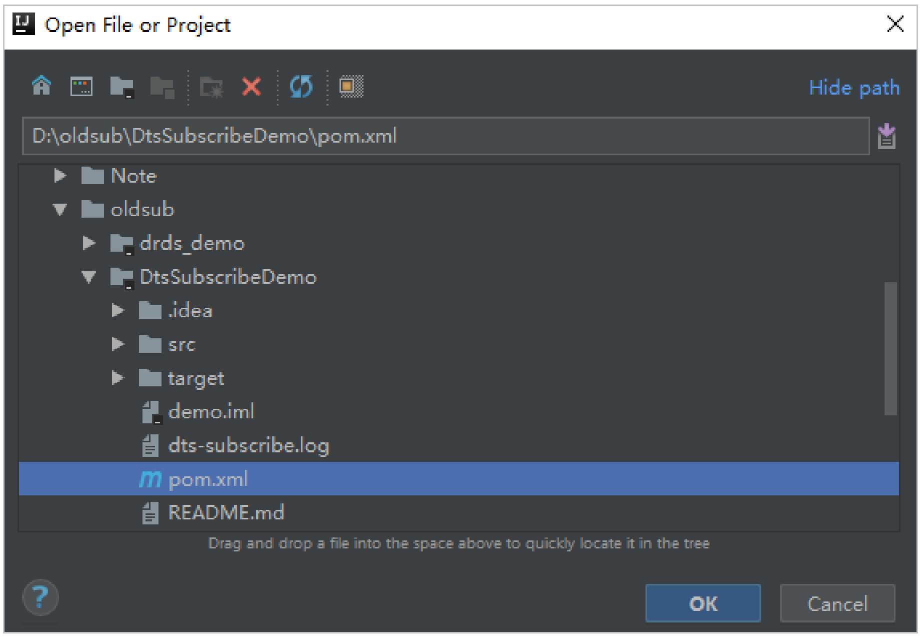
```
    <name>dts-new-subscribe-sdk</name>
    <url>https://www.aliyun.com/product/dts</url>
    <description>The Aliyun new Subscribe SDK for Java used for accessing Data Transmission Service</description>
    <packaging>jar</packaging>
    <groupId>com.aliyun.dts</groupId>
    <artifactId>dts-new-subscribe-sdk</artifactId>
    <version>1.3</version>
```

Notice You can obtain the latest version of the change tracking SDK from the Maven website. For more information, visit the Maven page of the change tracking SDK.

3. Open IntelliJ IDEA. In the window that appears, click **Open**.

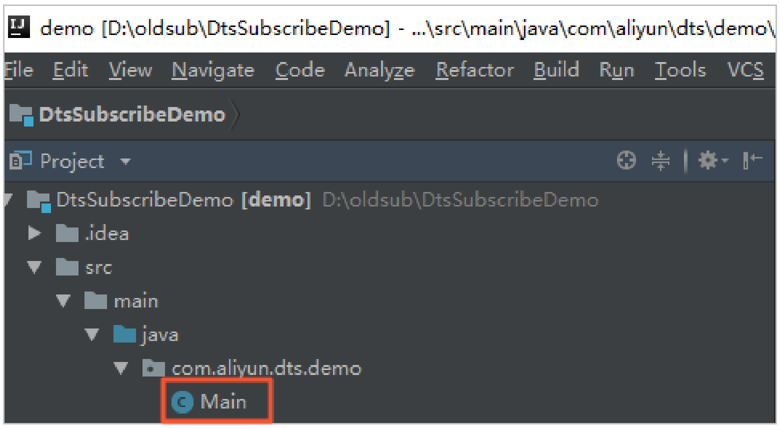


4. In the dialog box that appears, go to the directory where the package is decompressed, and expand the folders to find the *pom.xml* file.

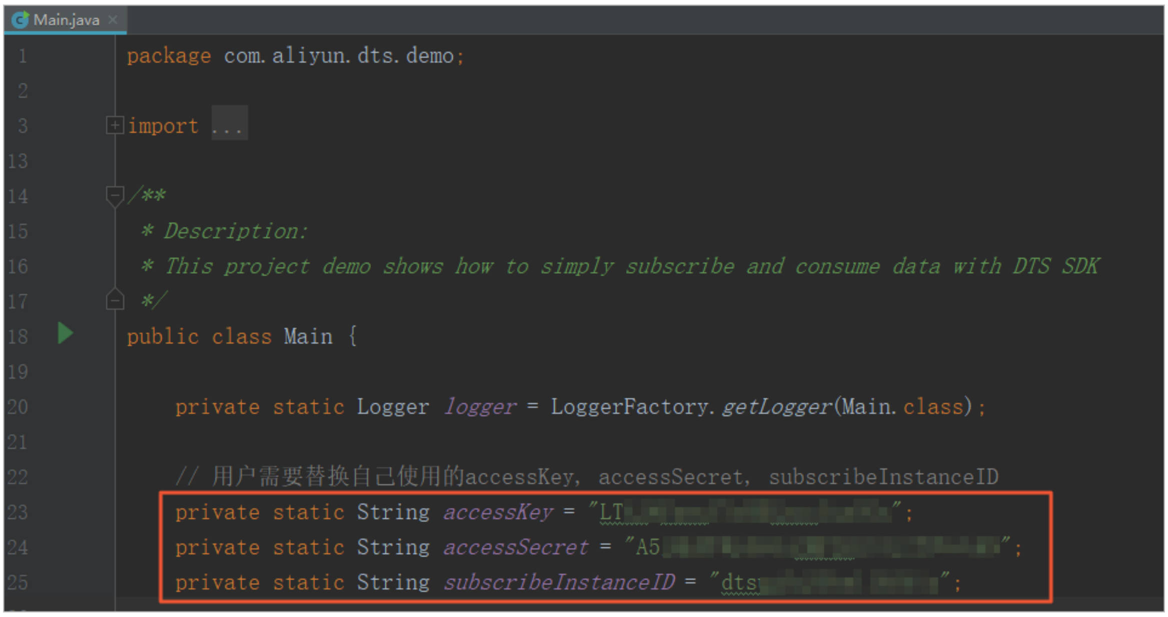


5. In the dialog box that is displayed, select **Open as Project**.

6. In the Project tool window of IntelliJ IDEA, expand the folders to find the demo file of the Java client, and double-click the file. The file name is *Main.java*.


demo [D:\oldsub\DtsSubscribeDemo] - ...\src\main\java\com\aliyun\dts\demo\
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS
DtsSubscribeDemo
Project
DtsSubscribeDemo [demo] D:\oldsub\DtsSubscribeDemo
.idea
src
main
java
com.aliyun.dts.demo
Main

7. Set the parameters for the demo file of the Java client.



```
package com.aliyun.dts.demo;

import ...

/**
 * Description:
 * This project demo shows how to simply subscribe and consume data with DTS SDK
 */
public class Main {

    private static Logger logger = LoggerFactory.getLogger(Main.class);

    // 用户需要替换自己使用的accessKey，accessSecret，subscribeInstanceID
    private static String accessKey = "LT[redacted]";
    private static String accessSecret = "A5[redacted]";
    private static String subscribeInstanceID = "dts[redacted]";
```

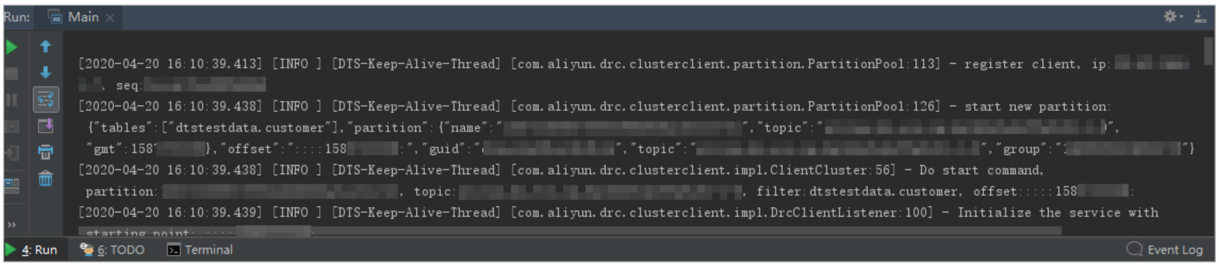
Parameter	Description
accessKey	Enter the AccessKey ID and AccessKey secret of your Alibaba Cloud account. For more information, see Before you begin.
accessSecret	
subscribeInstanceID	Enter the ID of the change tracking instance. To obtain the ID of the change tracking instance, perform the following steps: Log on to the DTS console. In the left-side navigation pane, click **Change Tracking**.

8. In the top menu bar of IntelliJ IDEA, choose **Run** > **Run** to run the client.

> **Note** The first time you run IntelliJ IDEA, a specific time period is required to load and install the relevant dependency.

Result

The following figure shows that the Java client can track data changes from the source database.



```
[2020-04-20 16:10:39.413] [INFO ] [DTS-Keep-Alive-Thread] [com.aliyun.drc.clusterclient.partition.PartitionPool:113] - register client, ip:[redacted], seq:[redacted]
[2020-04-20 16:10:39.438] [INFO ] [DTS-Keep-Alive-Thread] [com.aliyun.drc.clusterclient.partition.PartitionPool:126] - start new partition:
 {"tables":["dtstestdata.customer"],"partition":{"name":"[redacted]","topic":"[redacted]",
 "gmt":158[redacted]},"offset":"::::158[redacted]:","guid":"[redacted]","topic":"[redacted]","group":"[redacted]"}
[2020-04-20 16:10:39.438] [INFO ] [DTS-Keep-Alive-Thread] [com.aliyun.drc.clusterclient.impl.ClientCluster:56] - Do start command,
 partition:[redacted], topic:[redacted], filter:dtstestdata.customer, offset::::158[redacted]:
[2020-04-20 16:10:39.439] [INFO ] [DTS-Keep-Alive-Thread] [com.aliyun.drc.clusterclient.impl.DrcClientListener:100] - Initialize the service with
 starting point: ::::[redacted]
```

4.9. Download a change tracking SDK

This topic provides the links to download all available versions of change tracking SDKs, and describes the updates of these versions.

> **Note** You can obtain the latest version of the change tracking SDK from the Maven website. For more information, visit the Maven page of the change tracking SDK.

Release date	SDK version	Description
April 20, 2018	dts-subscribe-sdk-4.8.3.12.3.jar	The error messages of logs are optimized.
October 12, 2017	dts-subscribe-sdk-4.8.3.12.1.jar	• The tracked data can be transferred by using the binary format. This improves the performance of change tracking. • Bugs are fixed.
May 9, 2017	dts-subscribe-sdk-4.6.27.12.0.jar	Maven dependency is provided by this version and later. The following bugs are fixed: • Duplicate data is tracked if multiple connections are created for an SDK. • The SDK fails to submit the disaster recovery checkpoint. • When you call the `client.stop()` method to stop a thread, the thread does not stop as expected. • The `keep alive error` error occurs. • The `java.lang.NullPointerException` error occurs at line 152.
April 11, 2017	dts-subscribe-20170411.jar	The following bugs are fixed: The data pulling thread and data consumption thread are separated, and the change tracking client exits due to an exception.
April 26, 2016	dts-subscribe-3.0.jar	The first version of the change tracking client. Data changes from an ApsaraDB RDS for MySQL instance can be tracked.

References

Use the SDK demo code to consume tracked data