Informatica® Data Ingestion and Replication
August 2024

Database Ingestion and Replication

Informatica Data Ingestion and Replication Database Ingestion and Replication
August 2024

Publication Date: 2024-08-07

Table of Contents

Preface

Read *Database Ingestion and Replication* to learn how to configure database ingestion and replication tasks in the Data Ingestion and Replication service.

This publication also covers supported sources and targets, use cases, source preparation, limitations and usage considerations, deploying tasks, and running and managing database ingestion jobs.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Intelligent Cloud Services web site

You can access the Informatica Intelligent Cloud Services web site at http://www.informatica.com/cloud. This site contains information about Informatica Cloud integration services.

Informatica Intelligent Cloud Services Communities

Use the Informatica Intelligent Cloud Services Community to discuss and resolve technical issues. You can also find technical tips, documentation updates, and answers to frequently asked questions.

Access the Informatica Intelligent Cloud Services Community at:

https://network.informatica.com/community/informatica-network/products/cloud-integration

Developers can learn more and share tips at the Cloud Developer community:

https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers

Informatica Intelligent Cloud Services Marketplace

Visit the Informatica Marketplace to try and buy Data Integration Connectors, templates, and mapplets:

https://marketplace.informatica.com/

Data Integration connector documentation

You can access documentation for Data Integration Connectors at the Documentation Portal. To explore the Documentation Portal, visit https://docs.informatica.com.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Intelligent Cloud Services Trust Center

The Informatica Intelligent Cloud Services Trust Center provides information about Informatica security policies and real-time system availability.

You can access the trust center at https://www.informatica.com/trust-center.html.

Subscribe to the Informatica Intelligent Cloud Services Trust Center to receive upgrade, maintenance, and incident notifications. The Informatica Intelligent Cloud Services Status page displays the production status of all the Informatica cloud products. All maintenance updates are posted to this page, and during an outage, it will have the most current information. To ensure you are notified of updates and outages, you can subscribe to receive updates for a single component or all Informatica Intelligent Cloud Services components. Subscribing to all components is the best way to be certain you never miss an update.

To subscribe, on the Informatica Intelligent Cloud Services Status page, click **SUBSCRIBE TO UPDATES**. You can choose to receive notifications sent as emails, SMS text messages, webhooks, RSS feeds, or any combination of the four.

Informatica Global Customer Support

You can contact a Global Support Center through the Informatica Network or by telephone.

To find online support resources on the Informatica Network, click **Contact Support** in the Informatica Intelligent Cloud Services Help menu to go to the **Cloud Support** page. The **Cloud Support** page includes system status information and community discussions. Log in to Informatica Network and click **Need Help** to find additional resources and to contact Informatica Global Customer Support through email.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at https://www.informatica.com/services-and-training/support-services/contact-us.html.

CHAPTER 1

Database Ingestion and Replication

Database Ingestion and Replication can ingest and replicate data at scale from relational databases to multiple types of targets, including cloud data warehouses, data lakes, streaming systems, and other relational databases. Database Ingestion and Replication is a feature of the Cloud Data Ingestion and Replication service.

Database Ingestion and Replication provides an easy-to-use interface for configuring and deploying database ingestion and replication tasks. You can start the task wizard from the **Ingest** panel on the Data Integration unified Home page or from the **New** option in the navigation bar of the Home page. A deployed task is called a job or job instance. To run and monitor database ingestion and replication jobs, you can use any of the following monitoring interfaces: the **My Jobs** page accessed from the navigation bar on the Home page, the **All Jobs** and **Running Jobs** pages in the Monitor service, or the **Overview** and **All Jobs** pages in Operational Insights.

Database Ingestion and Replication can perform the following types of data load operations:

- *Initial load*. Loads source data read at a single point in time to a target in batch mode. After the data is loaded, the ingestion job ends. You can use this load type to materialize a target to which incremental changes will be sent later, to migrate from an on-premises database system to a cloud-based system, or to add data to a data lake or data warehouse.
- *Incremental load*. Replicates data changes continuously or until the job is stopped or ends. The job replicates the changes that have occurred since the last time it ran or from a specific start point. You can use this load type to keep data in separate reporting and analytics systems up to date so that you can make informed decisions for your business or organization based on the latest data. You can also use this load type to feed the latest changes to data warehouses and cloud data lakes for big data processing.
- *Initial and incremental load*. Performs an initial load of point-in-time data to the target and then automatically switches to replicating incremental data changes made to the same source tables on a continuous basis.

For more information about the sources and targets supported for each load type, see "Database Ingestion and Replication source and target types" on page 10.

Note: A database ingestion and replication task automatically maps source tables and fields to target tables and fields based on name matching. You can define rules to customize the target table names.

Use cases

Database Ingestion and Replication can be used to solve multiple business problems.

You can use Database Ingestion and Replication in the following scenarios:

- **Offline reporting**. Move user reporting activity from a mission-critical production database system to a separate reporting system to avoid degrading database performance.
- **Data warehousing**. Help build out data warehouses by transferring data from multiple databases, including on-premises databases, to the data warehouse system. After an initial batch load of data to the data warehouse, Database Ingestion and Replication can propagate data changes continuously from a source database to keep the data in the data warehouse up to date.
- **Real-time fraud detection**. Run real-time fraud detection analytics against a replica database that Database Ingestion and Replication keeps up to date by providing change data continuously. Fraud detection processes can then run against the latest data without degrading the source system.
- **Coordination with big data applications**. Keep data lakes synchronized with on-premises sources in a database management system (DBMS) or provide data to Data Integration for at scale processing.
- **Migration to cloud-based systems**. Migrate data from on-premises database systems to cloud-based systems.

Database Ingestion and Replication source and target types

The source and target types that Database Ingestion and Replication supports depend on the load type that the database ingestion and replication tasks use: an initial load point-in-time operation, an incremental load of only the data changes, or an initial load followed by an incremental load.

For supported source and target versions, see the KB article
What are the supported sources and targets for IICS Cloud Mass Ingestion service?.

Source types

The following table shows the source types that are supported (S) for each load type:

Source Type	Initial Load	Incremental Load	Initial and Incremental Loads
Db2 for i	S	S	S
Db2 for Linux, UNIX, and Windows (LUW)	S	S[1]	S[1]
Db2 for z/OS	S	S	S
Microsoft SQL Server, including on-premises SQL Server, RDS for SQL Server, Microsoft Azure SQL Database, and Azure SQL Managed Instance	S	S[2]	S[2]
MongoDB	S	S	-

Source Type	Initial Load	Incremental Load	Initial and Incremental Loads
MySQL, including on-premises MySQL, Amazon Aurora MySQL, Azure Database for MySQL[3], Cloud SQL for MySQL, and RDS for MySQL	S	S	S
Netezza	S	-	-
Oracle, including on-premises Oracle and RDS for Oracle	S	S	S
PostgreSQL, including on-premises PostgreSQL, Amazon Aurora PostgreSQL, Azure Database for PostgreSQL - Flexible Server, Cloud SQL for PostgreSQL, and RDS for PostgreSQL	S	S	S
SAP HANA, SAP HANA Cloud	S	S	-
Teradata	S	-	-

1. For Db2 for LUW sources, incremental load and combined load jobs are supported with the Query-based capture method only.
2. For Azure SQL Database sources, incremental load and combined load jobs are supported with the Query-based and CDC Tables capture methods. Log-based capture with a transaction log is not supported.
3. Azure Database for MySQL is supported with Snowflake targets for all load types and with Confluent Kafka targets for incremental loads.

To determine the connectors to use for these source types, see *Connectors and Connections > Database Ingestion and Replication connectors*.

Target types

The following table shows the target types that are supported (S) for each load type:

Target Type	Initial Load	Incremental Load	Initial and Incremental Loads
Amazon Redshift, Amazon Redshift Serverless	S	S	S
Amazon S3	S	S	S
Apache Kafka, Confluent Kafka, Amazon Managed Streaming for Apache Kafka (MSK)	-	S	-
Azure Event Hubs enabled for use with Kafka clients	-	S	-
Microsoft SQL Server, including on-premises SQL Server, RDS for SQL Server, Microsoft Azure SQL Database, and Azure SQL Managed Instance	S	S	S
Databricks	S	S	S
Flat file	S	-	-
Google BigQuery	S	S	S

Target Type	Initial Load	Incremental Load	Initial and Incremental Loads
Google Cloud Storage	S	S	S
Microsoft Azure Data Lake Storage Gen2	S	S	S
Microsoft Azure Synapse Analytics	S	S	S
Microsoft Fabric OneLake	S	S	S
Oracle, including RDS for Oracle	S	S	S
Oracle Cloud Infrastructure (OCI) Object Storage	S	S	S
PostgreSQL, including Amazon Aurora PostgreSQL and RDS for PostgreSQL	S	S	S
Snowflake	S	S	S

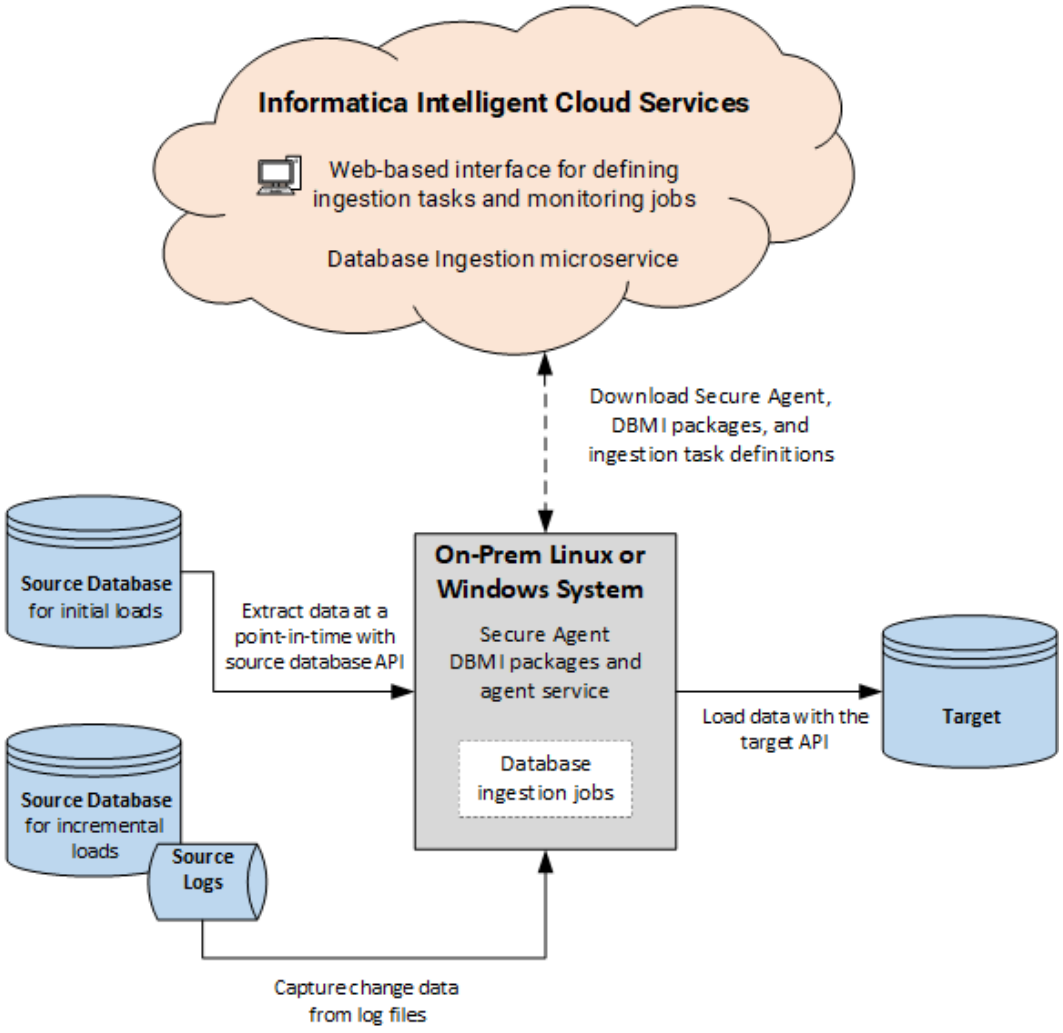
To determine the connectors to use for these target types, see *Connectors and Connections* > *Database Ingestion and Replication connectors*.

Database Ingestion and Replication architecture

Database Ingestion and Replication runs on a Secure Agent. You must install the Secure Agent on a Linux or Windows machine prior to using Database Ingestion and Replication.

After you start the Secure Agent the first time, the Database Ingestion and Replication agent and packages are installed locally. You then can use the Data Ingestion and Replication to configure database ingestion and replication tasks and to run and monitor database ingestion and replication jobs.

The following image shows the general architecture of Database Ingestion and Replication:



From the Web-based interface in Informatica Intelligent Cloud Services, you can create and manage ingestion tasks and run and monitor ingestion jobs.

The following interactions occur:

1. When you download the Secure Agent to your on-premises system, the Database Ingestion DBMI packages are also downloaded, provided that you have a license for Database Ingestion. You can then configure the DBMI agent service.
2. From the Informatica Intelligent Cloud Services Web-based interface, you define database ingestion and replication tasks.
3. When you deploy a data ingestion task, a corresponding executable job is created on the Secure Agent system.
4. When you run a database ingestion and replication job, the ingestion and replication task metadata is pushed down to the Secure Agent. The job uses this information to process data.
   - For an initial load operation, the database ingestion and replication job extracts data at a specific point-in-time from the source tables and fields. The job uses the database API of the relational source to retrieve the data.

- For an incremental load operation, the database ingestion and replication job captures changes, such as inserts, updates, and deletes, for the source tables and fields from the source database logs. Change data capture runs continuously or until the job is stopped or ends.

The data is loaded to the target using the appropriate target API.

Database Ingestion and Replication system requirements

The following table lists the Database Ingestion and Replication minimum system requirements for the Secure Agent:

Component	Minimum requirement
Cores per CPU	8 minimum, 16 recommended if you need to process a large number of source tables in an initial load
Memory	32 GB
Disk space	5 GB per job, based on a row size of 2 KB

Database Ingestion and Replication general limitations and guidelines

Before you configure database ingestion and replication tasks for initial, incremental, or combined initial and incremental load operations, review the following limitations and guidelines:

- Database Ingestion and Replication does not support the Cloud Hosted Agent, a runtime environment on a cloud platform, or a serverless runtime environment. Use a local Secure Agent installation.
- If a Secure Agent in a runtime environment that is assigned to a database ingestion and replication task terminates, you cannot undeploy the associated ingestion and replication job, update the task to specify another runtime environment, and deploy the task again. In this situation, perform one of the following actions:
  - Assign a different Secure Agent to the runtime environment. Ensure that the new Secure Agent is running. Then restart the associated database ingestion and replication job.
  - Copy the task. In the task copy, specify another runtime environment that has an active Secure Agent. Then deploy the task and run the associated database ingestion and replication job.
- Database Ingestion and Replication uses UTF-8 as the code page. If you select another code page type when defining a connection, Database Ingestion and Replication ignores it.
- Database Ingestion and Replication assumes that each row in a source table is unique. Informatica recommends that source tables have primary keys. If a table doesn't have a primary key, Database Ingestion and Replication uses the values in all columns, except LOB columns, to uniquely identify each source row. In this case, each Update operation is processed as a Delete followed by an Insert on the target.

Note: If you change the source primary key and have an Amazon S3, Flat File, Google Cloud Storage, or Microsoft Azure Data Lake Storage target, Database Ingestion and Replication processes each Update operation as a Delete followed by an Insert on the target.

- Database Ingestion and Replication uses the minimum source constraints needed for replication when generating target objects. If a source has a primary key, the job preferentially uses that primary key and no other constraint when creating the target object. If a source does not have a primary key but does have unique indexes, the job chooses the best available unique index for replication, or if all indexes are equal for replication purposes, the job chooses one at random. Only a single primary key or unique index is used. Also, source object properties that are not required for replication, such as for storage and partitioning, are not reflected on the target.
- Because Data Ingestion and Replication seeks to maintain consistency between the source and target when replicating data, it does not add or recognize any constraints on the target, such as those for nullability, foreign keys, and defaults. Doing so could lead to inconsistencies between the source and target and potential replication errors. For example, errors could occur if the NOT NULL constraint is added to a target column that receives data from a nullable source column, or if the DEFAULT constraint is added to a target column that receives source data that doesn't match the default.

  Note: If source constraints exist at the time of task deployment, they're not re-created at the target because they might not have existed previously when data was added to a source column and captured.
- If you run multiple incremental load or combined initial and incremental load jobs against the same source database, you cannot move capture processing of some source tables from one job to another job or merge the jobs into a single job, without resynchronizing the source and target tables being processed by the new job. Each job maintains a unique recovery checkpoint. If you try to move capture processing of a table from one job to another job, the correct recovery information for that table is not maintained, potentially causing data loss or duplicate events on the target.
- The log file names generated for initial load jobs are the combination of task name, schema name, and table name. If the resulting file name exceeds the maximum length allowed by your Secure Agent operating system, the file might not be created or might not be found. If you suspect that the file name could exceed the limits of your operating system, use a shorter task name to reduce the overall expected file name length.
- If a combined initial and incremental load job captures an incremental insert change record during the initial unload phase, the job manufactures a delete for the same row to remove any duplicate that might have been obtained from the initial unload. This manufactured activity will be reflected in audit and soft delete apply modes.
- Database Ingestion and Replication does not display error messages that are longer than 1024 characters in the user interface. Instead, Database Ingestion and Replication prompts you to view the log file with the error, which is automatically downloaded.
- If a source column has a numeric data type that is not compatible with any numeric data type on the target, Database Ingestion and Replication maps the source column to a target varchar column.

Database Ingestion and Replication sources - preparation and usage

Before you configure database ingestion and replication tasks for initial load, incremental load, or combined initial and incremental operations, prepare the source database and review any usage considerations for your sources to avoid unexpected results.

Db2 for i sources

To use Db2 for i sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations.

Source preparation

- For incremental load jobs and combined initial and incremental load jobs, grant the appropriate level of authority for accessing the Db2 journal and file objects that Database Ingestion and Replication uses to the user ID that runs database ingestion and replication jobs. The following table lists these objects and their Db2 authority requirements:

Object	Authority
Journal	*USE *OBJEXIST
Journal Library	*USE
Journal Receiver	*USE
Journal Receiver Library	*USE
File	*USE
File Library	*USE

- For incremental load jobs, journaling must be active on each database physical file that corresponds to a selected source table. Also, each journal must be configured with the IMAGES(*BOTH) option to store both before and after images of change data.

  If journaling is not active on a physical file for a source table, when you define a database ingestion and replication task, you can generate a CDC script that activates it. The script issues the following command, which activates journaling and sets the IMAGES option to BOTH:

  ```
  CALL QSYS2.QCMDEXC('STRJRNPF FILE(library/physical-file) JRN(library/journal-name)
  IMAGES(*BOTH)')
  ```

  If journaling is already active for a physical file for a source table, the CDC script output contains the following comment:

  ```
  Table 'table_name' is skipped because journaling is already enabled.
  ```

- By default, Database Ingestion and Replication uses the DataDirect JDBC for IBM Db2 driver to connect to the Db2 for i database. Informatica recommends that the first user who creates and tests a Db2 for i connection to the source database has DBA authority on the database. This authority is needed for the driver to create and upload the packages that it uses for Db2 access and to grant the EXECUTE privilege on the packages to PUBLIC. If a DBA user does not perform the first connection test, you must grant *USE authority on the CRTSQLPKG command for creating the packages and grant *CHANGE authority on the library in which the packages are created.
- To use SSL data encryption for Db2 for i sources, when you configure the Db2 for i connection properties, select **JTOpen** in the **JDBC Driver** field and select **SSL** in the **Encryption Method** field.
  Also, add the required certificates to the Informatica Cloud Secure Agent JRE cacerts keystore in one of the following locations:

  For Linux:

  ```
  Secure Agent Directory\jdk\jre\lib\security\cacerts
  ```

For Windows:

Secure Agent Directory\apps\jdk*LatestVersion*\jre

After you add the certificates, restart the Secure Agent to ensure the changes are reflected in the agent services app-truststore.jks files for the latest instances.

For more information about adding certificates to the keystore, see HOW TO: Import certificates into Informatica Cloud Secure Agent JRE.

Usage considerations

- Because Database Ingestion and Replication expects each source table row to be unique, Informatica recommends that each source table have a primary key. Database Ingestion and Replication does not honor unique indexes in place of a primary key. If no primary key is specified, Database Ingestion and Replication treats all columns as if they are part of the primary key.
- When you define a database ingestion and replication task, on the **Source** page, specify a journal name that is associated with the source tables that are enabled for journaling.

  Important: Ensure that the case and spelling of the name in the **Journal Name** field and table selection rules match the journal and table name values in the Db2 source catalog.
- Schema drift options are available for Db2 for i sources in database ingestion and replication incremental load and combined initial and incremental load jobs.
  If you set the **Add Column** option to **Replicate** and then add a column with a default value to a Db2 for i source table, Database Ingestion and Replication adds the default value to the newly added table rows to the target. However, existing rows on the target are not updated to reflect the default values. To get the default value populated to the existing target rows, perform another initial load to re-materialize the target.
- Database Ingestion and Replication does not support the following Db2 for i data types:

  - BLOB
  - CLOB
  - DATALINK
  - DBCLOB
  - GRAPHIC
  - LONG VARGRAPHIC
  - VARGRAPHIC
  - XML

  Database ingestion and replication jobs propagate nulls for columns that have these data types.

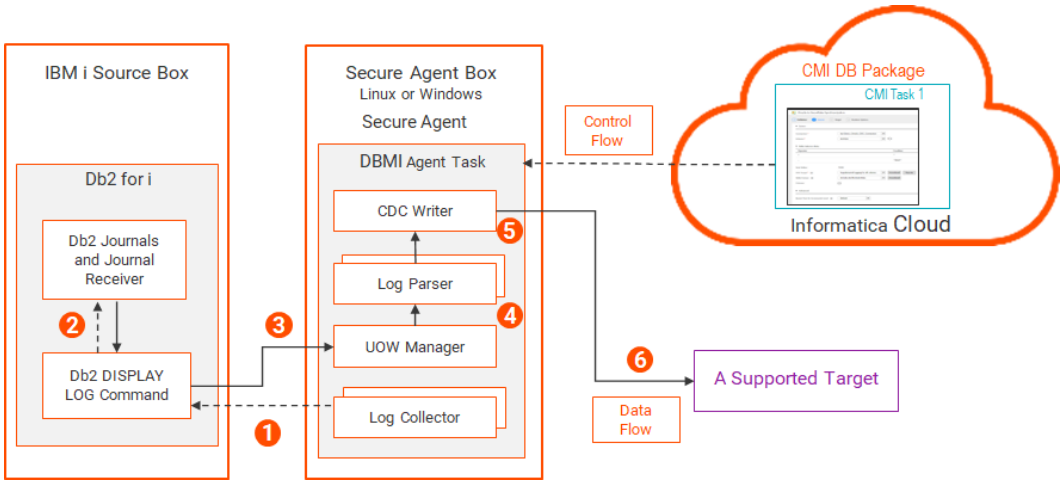
  For information about the default mappings of supported source data types to target data types, see "Default Data Type Mappings" on page 172.
- To prevent the system from deleting Db2 journal receivers while database ingestion and replication incremental load and combined initial and incremental load jobs are reading changes from the receivers, Database Ingestion and Replication provides a journal receiver exit program. The exit program locks the journal receivers during CDC processing. To use the exit program, you must install it and set a few custom properties. For more information, see "Database Ingestion and Replication journal receiver exit" on page 18 and "Installing the Database Ingestion and Replication journal receiver exit" on page 20.

Change capture mechanism for Db2 for i sources

Database Ingestion and Replication provides a single change capture mechanism and architecture for capturing change data from a Db2 source on IBM i and applying the data to a target.

The Secure Agent must run on a Linux or Windows box, separate from the IBM i source system. The network bandwidth between the IBM i system and the Secure Agent box must be robust. Informatica recommends network transfer rates that can handle100s of gigabits or one or more gigabytes of log data. If the network transfer rate is not capable of delivering the log data to the Secure Agent at speeds equal to or greater than the rate at which Db2 produces log data of CDC interest, the database ingestion job will not be able to provide the data to the target in a timely manner. If the data throughput does not meet your SLA, consider changing the hardware to increase the ethernet bandwidth between the IBM i system and the Secure Agent box.

The following image shows the Db2 for i change capture components and data flow:



1. The log collector, which runs in the DBMI Agent service under the Secure Agent, sends a request to the Db2 DISPLAY LOG command for Db2 for i journaled data.
   Each request includes a starting RBA along with a list of tables of CDC interest for database ingestion.
2. The command requests the data for the source tables from the Db2 journals and journal receiver.
3. The command returns the journal entries containing the data to the UOW (Unit Of Work) Manager.
4. The UOW Manager sends the journaled data in committed transaction order to the log parser.
5. The log parser parses the DML changes from the committed transactions. Before sending the data to the CDC writer, the log parser transforms it into a canonical form of the Db2 journal data that can be consumed by the DBMI Agent task and applied to the target.

   Note: Because this resource-intensive activity occurs on the Secure Agent box, CPU consumption on the IBM i system is minimized.
6. The CDC writer applies the formatted data to the target.

Database Ingestion and Replication journal receiver exit

Database Ingestion and Replication provides an exit program to prevent the deletion of Db2 for i journal receivers that are in use for change data capture processing. Use the journal receiver exit to prevent potential errors related to journal receiver deletion during CDC.

The exit program locks the journal receivers while they're being read by database ingestion and replication incremental load or combined load jobs. If the exit program finds a lock record for a journal receiver during

CDC processing, the exit program returns a response that prevents the system from deleting the journal receiver.

When the database ingestion and replication job switches to the next journal receiver on the chain, Database Ingestion and Replication removes the lock record for the journal receiver that was last read and adds a lock record for the next journal receiver to be read. The system will not remove any journal receivers that are created subsequent to the currently locked receiver.

The control information for the journal receiver exit is stored in a Db2 for i table on the source system. The table contains the following columns:

- JOURNAL_RECEIVER_LIBARY. The name of the journal receiver library.
- JOURNAL RECEIVER_NAME. The name of a journal receiver.
- DBMI_MAPPING_NAME. A unique lock token name.
- LOCKED_BY_USER. The user ID of the connected user for which the lock was acquired.
- TIME_LOCKED. The time at which the lock was created.

Each combination of JOURNAL_RECEIVER_LIBRARY and JOURNAL_RECEIVER_NAME entries identifies a journal receiver instance that will be locked so that it can't be deleted during change data capture.

To use the journal receiver exit, perform the following steps:

1. Install the journal receiver exit. The installation process registers the Delete Journal Receiver exit program at the QIBM_QJO_DLT_JRNRCV exit point. For more information, see "Installing the Database Ingestion and Replication journal receiver exit" on page 20.
2. Specify the following custom properties on the **Source** page of the task wizard, for each task with a Db2 for i CDC source:

Custom property	Description
pwx.cdcreader.iseries.option.useJournalReceiverExit	Set this property to true to enable the use of the journal receiver exit for job instances deployed from the task.
pwx.cdcreader.iseries.option.JournalReceiverExitJobToken	When pwx.cdcreader.iseries.option.useJournalReceiverExit is set to true, you must specify a token string up to 256 characters in length that is unique for each database ingestion and replication job instance. If the token string is not unique across all job instances, unpredictable results can occur, especially when multiple jobs are accessing the same journal receivers. Also, ensure that the token value remains the same after a job Resume or Redeploy operation.
pwx.cdcreader.iseries.option.useJournalReceiverQueries	Set this property to true to enable Database Ingestion and Replication to check the maintenance level on the Db2 machine to determine if it's capable of doing journal receiver queries.

Installing the Database Ingestion and Replication journal receiver exit

Use the Data Ingestion and Replication journal receiver exit to prevent the deletion of Db2 for i journal receivers while they're in use for change data capture. To manually install journal receiver exit, complete the following steps:

1. Unzip the V01_Exit_Install file to a local folder on your computer.
2. Run the SQL statements in the SQL_1.txt file to check if the system has enough space for the installation. A return value of 1 means the system has enough space.
3. Run the SQL statements in the SQL_2.txt file to create the schema and the default journal and journal receiver for journal receiver exit point handling.
4. Run the FTP commands in the FTP_3.txt file to install the journal receiver exit.

   Note: Before starting FTP, navigate to the directory with the IBMi_SaveFile_V01.savf file, which contains journal receiver exit program.
5. Run the SQL statements in the SQL_4.txt file to add the exit point to the IBM i system.

   Note: Before running the SQL, replace <userId> with a valid user ID for the system.
6. Use the SQL statements in the SQL_5.txt file to create the Db2 objects used for locking journal receivers to prevent them from being deleted.

Db2 for LUW sources

To use Db2 for Linux, UNIX, and Windows (LUW) sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations.

Source preparation

- Database Ingestion and Replication requires users to have the SELECT privilege on several system catalog tables and views. Use the following grant statement:

  ```
  GRANT SELECT ON <catalog_table> TO <dbmi_user>
  ```

  Issue this grant for each of the following catalog tables or views:

  - SYSCAT.COLUMNS
  - SYSCAT.DATAPARTITIONEXPRESSION
  - SYSCAT.DATAPARTITIONS
  - SYSCAT.INDEXCOLUSE
  - SYSCAT.INDEXES
  - SYSCAT.SCHEMATA
  - SYSCAT.TABLESSYSCAT.TABLES
  - SYSIBM.COLUMNS
  - SYSIBM.SYSDUMMY1
  - SYSIBM.SYSPLAN
  - SYSIBM.SYSVERSIONS
- To create and execute the Db2 package required for Database Ingestion and Replication to successfully connect to the Db2 database and issue SQL requests, ensure that the BINDADD privilege is granted to the *dbmi_user*:

  ```
  GRANT BINDADD ON DATABASE TO <dbmi_user>
  ```

Usage considerations

- Database ingestion and replication incremental load and combined initial and incremental load jobs that have a Db2 for LUW source must use the **Query-based** capture method. Query-based change data capture uses a SQL statement with a WHERE clause that references a common CDC query column to identify the rows with insert and update changes. Configuration of the source database is limited to adding the CDC query column to each source table. Users must have at least read only access to the source tables. The CDC query column type must be equivalent to timestamp, without a time zone. Currently, this CDC functionality has been tested with Snowflake targets only.
- Because Database Ingestion and Replication expects each source table row to be unique, Informatica recommends that each source table have a primary key. Database Ingestion and Replication does not honor unique indexes in place of a primary key. If no primary key is specified, Database Ingestion and Replication treats all columns as if they are part of the primary key.
- Database ingestion and replication jobs can replicate data from columns that have LOB data types to Microsoft Azure Data Lake Storage Gen 2, Microsoft Azure Synapse Analytics, or Snowflake targets. For information about the default mappings of supported source data types to target data types, see "Default Data Type Mappings" on page 172.

Db2 for z/OS sources

To use Db2 for z/OS sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations.

Source preparation

- Database ingestion and replication incremental load jobs with a Db2 for z/OS source use a stored procedure to call the Db2 Instrumentation Facility Interface (IFI) to read change data from Db2 logs on the z/OS source system. Database Ingestion and Replication delivers the stored procedure libraries and JCL in the Db2 for zOS Database Ingestion connector package ZIP file. You must receive the stored procedure libraries into an APF-authorized library and then customize the JCL for your environment. The stored procedure runs in a Workload Manager (WLM) address space on the Db2 source system. For more information about z/OS system requirements, stored procedure setup, and required permissions, see "Installing and configuring the stored procedure for Db2 for z/OS CDC" on page 22.
- When you define a Db2 source in a database ingestion and replication incremental load task, you must select the option **Enable CDC for all columns** in the **CDC Script** field. Database Ingestion and Replication generates a script for enabling Db2 DATA CAPTURE CHANGES on the source tables and on certain Db2 catalog tables that are used for CDC. After this attribute is set for one job, all other jobs recognize that the attribute is enabled in Db2 for the catalog tables needed. You can either execute the CDC script from the user interface if you have sufficient privileges or ask a Db2 DBA who has SYSDBA authority to run the script.
- Make sure the Database Ingestion and Replication user has the Db2 for z/OS privileges that are required for the database ingestion and replication load type to be performed. For more information, see "Db2 for z/OS privileges" on page 26.
- Database Ingestion and Replication uses the Progress DataDirect JDBC IBM DB2 driver to connect to a Db2 for z/OS source. For a new implementation of Database Ingestion and Replication with Db2 for z/OS sources, the user who first runs an incremental job must have SYSADM or SYSDBA authority to establish JDBC connectivity.

- If you perform both initial loads and incremental loads and want to use different users with different Db2 for z/OS privileges for these job types, perform the following steps:
  1. Create separate Db2 for z/OS Database Ingestion connections for initial loads and incremental loads. In the connection properties, specify a user who has the Db2 privileges required for an initial load job or incremental load job that uses the connection.
  2. To restrict access to connections to certain users, you can set explicit user permissions on a connection asset in Administrator. To use the connection for running a job, the user must have the Execute permission. This practice prevents a user who has a lower level of privileges from running a job that requires a higher level of permissions.
  3. When you create a database ingestion and replication that uses the initial load or incremental load type, select the connection that specifies a user with the required Db2 privileges and asset permission.

Usage considerations

- Because Database Ingestion and Replication expects each source table row to be unique, Informatica recommends that each source table have a primary key. Database Ingestion and Replication does not honor unique indexes in place of a primary key. If no primary key is specified, Database Ingestion and Replication treats all columns as if they are part of the primary key.
- The default setting for z/OS is that schema drift is enabled when you run the ALTER statements for the catalog tables. Once schema drift is enabled for one job, it is enabled for all jobs. If jobs are running when the ALTER statements are running, the running jobs do not have schema drift enabled until they are subsequently stopped and resumed.
- Database Ingestion and Replication does not support schema drift for Db2 11 for z/OS sources.
- Database Ingestion and Replication does not support the following Db2 for z/OS data types:

  - BLOB
  - CLOB
  - DBCLOB
  - XML

  Database ingestion and replication jobs propagate nulls for columns that have these data types.

  For information about the default mappings of data types, see "Default Data Type Mappings" on page 172.

Installing and configuring the stored procedure for Db2 for z/OS CDC

To perform Db2 for z/OS CDC processing for incremental load jobs, Database Ingestion and Replication provides a stored procedure that runs on the z/OS source system. The stored procedure calls the Db2 Instrumentation Facility Interface (IFI) to collect change data from the Db2 logs.

z/OS system requirements

Before you begin, verify that the Db2 for z/OS source system meets the following requirements:

- The z/OS system has the recommended operating system version of 2.3 or later.
- The Db2 for z/OS source uses Db2 version 11 or 12.
- Ensure that a Workload Manager (WLM) address space exists for executing the Database Ingestion and Replication stored procedure.

If you have not set up a Db2 WLM address space, see the following IBM documentation for more information:

- For Db2 12: https://www.ibm.com/support/knowledgecenter/en/SSEPEK_12.0.0/inst/src/tpc/db2z_setupwlmenvironment.html

- Ensure that the library where the stored procedure for Db2 for z/OS CDC will be received exists and is APF-authorized on the z/OS system.

Install the Stored Procedure Libraries and Customize the JCL

On your client machine, perform the following steps:

1. Verify that the Db2 for zOS Database Ingestion connector is available.

   The Db2 for zOS Database Ingestion connector package contains the Db2 for z/OS stored procedure libraries. When the connector is enabled for the Secure Agent Group (runtime environment), the connector package .zip file is downloaded to the downloads folder in the installation location. The package name has the format package-DB2ZMI.*nnnnn*, where *nnnnn* is an incremented package version number. If multiple package versions exist, use the latest one.

2. Unzip the package-DB2ZMI.*nnnnn*.zip file. The stored procedure files are added to the Db2WLMStoredProcedure folder under the package name.

3. Use FTP to transfer the #STPINST file in the Db2WLMStoredProcedure folder to a sequential file, PDS, or PDSE on the z/OS system.

   Notes:

   - Transfer the file without binary mode set.
   - Add a High Level Qualifier (HLQ) if needed to meet system requirements.

4. Use FTP or another file transfer method to transfer each of the following files to a separate data set on the z/OS system:

   - DBMI.ZOS.DBRMLIB.XMI
   - DBMI.ZOS.LOADLIB.XMI
   - DBMI.ZOS.USERJCL.XMI

   Notes:

   - Add or edit the HLQ if needed to meet system requirements.
   - Transfer the files in binary mode.
   - Ensure that each data set has the DCB attributes of LRECL=80, BLKSIZE=3120, and RECFM=FB.

     You might need to pre-allocate a data set to specify the required attribute values.

On the z/OS system, use TSO to receive the transmit (XMI) data sets into an APF-authorized library and then edit the stored procedure JCL member. Also, set Db2 user privileges and add a row to the resource limit table if used.

1. Receive the DBRMLIB transmit data set:

   ```
   RECEIVE INDATASET(DBMI.ZOS.DBRMLIB.XMI)
   ```

   Notes:

   - If you specified a HLQ when transferring the data set to z/OS, include the HLQ.
   - When you see the message `INMR906A Enter restore parameters or 'DELETE' or 'END' +`, enter your APF-authorized library:

     ```
     DA(your.library_name) UNIT(unit) VOLUME(volume)
     ```

The UNIT() and VOLUME() operands are optional. Include them if your installation does not put RECEIVE files on a work unit or volume by default.

2. Receive the LOADLIB transmit data set:

```
RECEIVE INDATASET(DBMI.ZOS.LOADLIB.XMI)
```

See the Notes in Step 1.

3. Receive the USERJCL transmit data set:

```
RECEIVE INDATASET(DBMI.ZOS.USERJCL.XMI)
```

See the Notes in Step 1.

4. Customize the #STPINST file that contains the stored procedure JCL for your environment.

The JCL creates the stored procedure and a global temporary table that holds the results of the requests to the Db2 IFI for data. It also binds the stored procedure package.

Notes:

- Based on the comments at the top of the JCL file, replace variables in the JCL with the values appropriate for your z/OS environment, including the Db2 subsystem name (!DSN!), stored procedure schema name (!SCHEMA!), stored procedure name (!STRPRC!), WLM environment name (!WLMENV!), and the name of the DBRMLIB transmit data set (!DBRMLIB! received in Step 1.
- If you used a HLQ for the received data sets, include the HLQ in the JCL.
- The WLM environment name is specified in the procedure APPLENV parameter or in the EXEC PARM of the WLM address space.
  In the procedure parameter:

```
//STARTING EXEC DSNBWLMG,DB2SSN=DSNB,APPLENV='DSNBWLM_GENERAL'
```

  In the EXEC PARM:

```
PARM='DSNB,40,DSNBWLM_GENERAL'
```

- You can use the received LOADLIB library after it is APF-authorized, or copy the contents of the library to your own APF-authorized library.
- The STEPLIB concatenation in the WLM address space must contain only APF-authorized libraries for the Db2 IFI to run.

5. Customize the JCL in members of the received USERJCL data set. For more information, see " Db2 for z/OS USERJCL Data Set " on page 24.
6. Ensure that the required Db2 privileges have been granted before you run the stored procedure JCL job. For more information, see "Db2 for z/OS privileges" on page 26.
7. Add a row in the Db2 DSNRLST*xx* resource limit table for database ingestion packages to ensure processor resources are adequate for stored procedure processing. Otherwise, incremental load jobs might abend. Add the row with the following columns:
   - An AUTHID column with the authentication ID that the database ingestion and replication task uses OR an RLFPDG column with the same package name as the database ingestion and replication task OR both of these columns.
   - An ASUTIME column defined with NULL or with a resource limit that is greater than the default limit.

   Then issue the Db2 -START RLIMIT command for For the changes to the resource limit table to take effect.

Db2 for z/OS USERJCL Data Set

The downloaded USERJCL data set is a partitioned data set (PDS) or extended partitioned data set (PDSE) that contains a JCL member for running a job that collects information that is not readily available otherwise when using Informatica-supplied WLM Db2 stored procedure.

You can fully customize member JCL in the USERJCL PDS or PDSE. Informatica recommends that you create a copy of a delivered JCL member under another name and then customize the copy so that original member is retained intact for reference.

If you install multiple stored procedures on multiple Db2 subsystems, you can install only one USERJCL library and create members in it that are tailored for each Db2 subsystem. Alternatively, you can create a separate library for a specific Db2 subsystem. Make sure that the database ingestion and replication job contains the correct library and member information for the Db2 subsystem.

The USERJCL PDS or PDSE contains the following member:

LOGINV member

LOGINV contains JCL for a job that obtains a Db2 log inventory list. If you set the **Initial Start Point for Incremental Load** property to **Specific Date and Time** for an ingestion task in the task wizard, the inventory list is used to determine the start point in the logs when the database ingestion and replication job runs for the first time or restarts. The log inventory list provides Database Ingestion and Replication with a starting RBA or LSN and an ending RBA or LRSN, along with the starting and ending timestamps of all active and archive logs for the Db2 subsystem. Database Ingestion and Replication uses this information to select the appropriate log archive to search for the requested start point. If the **Initial Start Point for Incremental Load** property is set to any option other than **Specific Date and Time**, you do not need to install the USERJCL library.

Contents of the LOGINV member:

```
//<USERID>I JOB  'LOG INVENTORY',MSGLEVEL=(1,1),MSGCLASS=X,
//      NOTIFY=&SYSUID,CLASS=A,REGION=0M
//* -----------------------------------------------------
//*
//*  PLEASE CHANGE DSN? TO THE DB2 SUBSYSTEM OF YOUR CHOICE.
//*
//*  THIS JCL MEMBER CAN BE USED TO RUN A LOG INVENTORY
//*  LIST. DBMI WILL REQUEST THIS IF A DBMI JOB IS TRYING TO
//*  RETART USING TIME. THE LOG INVERNTORY GIVES DBMI THE
//*  ABILITY TO CORRELATE RBA/LSRN TO ACTUAL LOG RESTART
//*  POSITIONS. DBMI PARSES THE SYSPRINT OUTPUT TO GET THE
//*  REQUIRED INFORMATION.
//*
//*
//*  SUBSTITUTION TAGS
//*  _________________
//*
//*  SUBSTITUTION TAGS ARE USED TO INSERT DATA INTO THE JOB
//*  BY DBMI BEFORE IT IS SUBMITTED. YOU MAY COPY THIS JCL
//*  INTO ANOTHER MEMBER AND MODIFY ITS CONTENTS. SUBSTITUTION
//*  TAGS MAY ALSO BE REMOVED AND HARD CODED FOR INDIVIDUAL
//*  JOB NEEDS.
//*
//*
//*  <USERID>     WILL BE REPLACED WITH THE USER ID.
//*
//* -----------------------------------------------------
//LOGINV EXEC PGM=DSNJU004
//STEPLIB DD DISP=SHR,DSN=DSN?10.SDSNLOAD
//        DD DISP=SHR,DSN=DSN?10.DSNC.RUNLIB.LOAD
//SYSUT1   DD DISP=SHR,DSN=DSN?.BSDS01
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//
```

Note: In this JCL, Database Ingestion and Replication replaces <USERID> with the user specified in the Db2 for z/OS Database Ingestion connection properties

To submit USERJCL member jobs and retrieve output from submitted jobs, Database Ingestion and Replication executes Db2-supplied stored procedures in batch as required. Make sure that the Database Ingestion and Replication user has the required privileges to execute these stored procedures. For more

information about Db2-supplied procedures, see https://www.ibm.com/docs/en/db2-for-zos/12?topic=sql-procedures-that-are-supplied-db2.

A USERJCL member is processed in the following manner:

1. For the LOGINV member, if the **Initial Start Point for Incremental Load** advanced property is set to **Specific Date and Time** on the **Source** page of the database ingestion and replication task wizard, Database Ingestion and Replication checks whether the following custom properties are specified for the source:
   - **pwx.cdcreader.ZOS.Db2JobsDSN**, which specifies the name of the installed USERJCL PDS or PDSEd
   - **pwx.cdcreader.ZOS.Db2JobLLOGINVMember**, which specifies the LOGINV member name that will be used for a database ingestion and replication job.
2. Database Ingestion and Replication reads the LOGINV member by using the Db2-supplied stored procedure ADMIN_DS_BROWSE.
3. Database Ingestion and Replication substitutes any tags required for the job execution.
4. Database Ingestion and Replication submits the job to the Db2 for z/OS database by using the Db2-supplied stored procedure ADMIN_SUBMIT_JOB.
5. Database Ingestion and Replication queries the status of the submitted jobs by using the Db2-supplied stored procedure ADMIN_JOB_QUERY.
6. After the job completes, Database Ingestion and Replication obtains the job output by using the Db2-supplied stored procedure ADMIN_JOB_FETCH.

Db2 for z/OS privileges

To deploy and run a database ingestion and replication task that has a Db2 for z/OS source, the source connection must specify a Database Ingestion and Replication user who has the privileges required for the ingestion load type.

Privileges for initial load processing

- For initial load jobs, grant the following privilege to the user specified in the Db2 for zOS Database Ingestion connection properties:

  `SELECT on schema.table TO user`

  Where *schema.table* identifies a source table.
- To run queries against Db2 catalog tables during source unload processing, grant the SELECT privilege on the following catalog tables:

  - SYSIBM.SYSCOLUMNS
  - SYSIBM.SYSINDEXES
  - SYSIBM.SYSKEYS
  - SYSIBM.SYSTABLEPART
  - SYSIBM.SYSTABLES
  - SYSIBM.SYSTABLESPACE

Privileges for incremental load processing

For incremental load jobs, grant the following privileges.

Note: If you do not have the authority required to issue these grants, ask a Db2 administrator who has SYSDBA authority or a higher authority level to issue them.

- To generate the CDC script that enables the database CDC option from the user interface, issue the following grant to the interface user:

```
GRANT ALTER TABLE schema.table  DATA CAPTURE CHANGES TO user   <--for each source table
 COMMIT;
```

- For incremental change data processing, grant the following privileges to the user specified in the Db2 for zOS Database Ingestion connection properties:
  - To make the change data in the Db2 logs available to the stored procedure:

```
GRANT ALTER TABLE schema.table  DATA CAPTURE CHANGES TO user   <--for each source
table
COMMIT;
```

  - To enable the user to obtain required information from the Db2 Instrumentation Facility Interface (IFI):

```
GRANT MONITOR2 TO user;
COMMIT;
```

  - To grant authorities on the global temporary table to the user:

```
GRANT READ ON !SCHEMA!.!STRPRC!_RS_TBL to user;
GRANT DELETE ON !SCHEMA!.!STRPRC!_RS_TBL to user;
GRANT EXECUTE !SCHEMA!.!STRPRC! to user;
COMMIT;
```

    Where !SCHEMA! and !STRPRC! are variables in the job JCL, which represent the stored procedure schema name and procedure name respectively.

    The first two privileges allow the user to read and delete the contents of the Global Temporary Table to which the stored procedure is writing the log data. The third privilege allows the user to run the stored procedure.

  - To enable the stored procedure to bind its Db2 plan, grant the following privileges to PUBLIC:

```
GRANT BIND, EXECUTE ON PLAN !STRPRC! TO PUBLIC;
COMMIT;
```

  - To run queries against Db2 catalog tables during source unload processing, grant the SELECT privilege on the following catalog tables:
    - SYSIBM.SYSCOLUMNS
    - SYSIBM.SYSDUMMY1
    - SYSIBM.SYSINDEXES
    - SYSIBM.SYSKEYS
    - SYSIBM.SYSTABLEPART
    - SYSIBM.SYSTABLES
    - SYSIBM.SYSTABLESPACE

Permissions Required for Executing the Stored Procedure on z/OS

Ensure that the following Db2 permissions are granted before you run the stored procedure JCL job:

- Ensure that the user who executes the stored procedure job has SYSADM authority, or ask the Db2 for z/OS DBA to run it.
- For the stored procedure to run, you must grant the following Db2 permissions to the procedure schema name specified in the #STPINST JCL file:
  - SELECT authority on Db2 catalog tables:

```
GRANT SELECT ON SYSIBM.* TO schema;
```

- EXECUTE authority on the package name specified in the JCL.

```
GRANT EXECUTE ON PACKAGE package_name TO schema;
```

Additionally, grant INSERT and DELETE authority on the schema and stored procedure name that are specified in the JCL for the global temporary table:

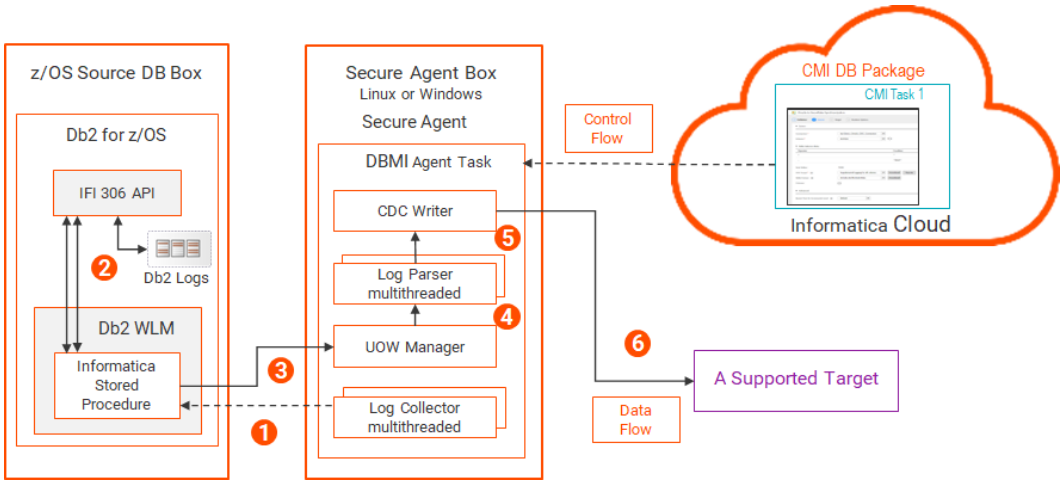
```
GRANT INSERT, DELETE ON schema.stored_procedure_name_RS_TBL TO user
```

Change capture mechanism for Db2 for z/OS sources

Database Ingestion and Replication provides a single change capture mechanism and architecture for capturing change data from a Db2 source on z/OS and applying the data to a target. This architecture uses multithreaded processing to optimize the performance of collecting data and of parsing and transforming the data into a format the target accepts.

The Secure Agent must run on a Linux or Windows box, separate from the Db2 z/OS source system. The network bandwidth between the z/OS system and the Secure Agent box must be robust. Informatica recommends network transfer rates that can handle100s of gigabytes or one or more gigabits of log data. If the network transfer rate is not capable of delivering the log data to the Secure Agent at speeds equal to or greater than the rate at which Db2 produces log data of CDC interest, the database ingestion and replication job will not be able to provide the data to the target in a timely manner. If the data throughput does not meet your SLA, consider changing the hardware to increase the ethernet bandwidth between the z/OS system and the Secure Agent box.

The following image shows the Db2 for z/OS change capture components and data flow:



1. The multithreaded log collector, which runs in the DBMI Agent service under the Secure Agent, issues multiple concurrent requests for Db2 log data to the Db2 stored procedure.
   Each request includes a starting RBA, or a starting LRSN if the logs are in a data-sharing environment, along with a list of tables of CDC interest for database ingestion.

   While processing a sequence of log data, the log collector can request the next sequence of log data.
2. The Db2 Instrumentation Facility Interface (IFI) API extracts data for the selected source tables of CDC interest from active and archive logs. The IFI then transfers the data, in raw, native form, to the Informatica Db2 stored procedure in z/OS Workload Manager (WLM).
3. The Db2 stored procedure returns the UOWs with captured data to the UOW Manager.
4. The UOW Manager sends the UOWs in commit order to the log parser.
5. The multithreaded log parser concurrently parses DML changes from the committed UOWs. The result is raw native Db2 log data in the format that the target type expects.

Note: Because this resource-intensive activity occurs on the Secure Agent box, CPU consumption on the z/OS system is minimized.

6. The CDC writer applies the formatted data to the target.

Microsoft SQL Server, RDS for SQL Server, Azure SQL Database, and Azure Managed Instance sources

To use Microsoft SQL Server sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations. SQL Server source types include on-premises SQL Server, Relational Database Service (RDS) for SQL Server, Azure SQL Database, and Azure SQL Managed Instance.

Source preparation

- For your SQL Server source type, verify that you use a SQL Server edition and version that Database Ingestion and Replication supports. See the KB article FAQ: What are the supported sources and targets for IICS Cloud Mass Ingestion service?.
- For incremental change data capture (CDC) operations with SQL Server sources, Database Ingestion and Replication provides multiple change capture methods. The preparation of your SQL Server source database depends on the CDC method you use.
  The available change capture methods are:
  - Log-based change data capture with the transaction log and CDC tables. Database Ingestion and Replication reads data changes from the SQL Server transaction log and the enabled SQL Server CDC tables. This method requires users to have extended privileges. SQL Server CDC must be enabled on the source tables.
  - Change data capture with CDC tables only. Users must have at least SELECT permission on the source and CDC tables. SQL Server CDC must be enabled on the source tables.
  - Query-based change data capture. Change data capture uses a SQL statement with a WHERE clause that references a common CDC query column to identify the rows with insert and update changes. Configuration of the source database is limited to adding the CDC query column to each source table. Users must have at least read only access to the source tables.

  For more information, see “Change capture mechanisms for SQL Server sources” on page 33.
- For incremental load jobs that use log-based CDC with transaction logs, ensure that the database user you specify in the SQL Server source connection has the db_owner role and the VIEW ANY DEFINITION privilege. To grant these privileges, use one of the following sets of SQL statements, depending on your SQL Server source type.
  For SQL Server on-premises sources, including Azure SQL Managed Instance:

```
USE master;
CREATE DATABASE <database>;
CREATE LOGIN <login_name> WITH PASSWORD = '<password>';
CREATE USER <user> FOR LOGIN <login_name>;
GRANT SELECT ON master.sys.fn_dblog TO <user>;
GRANT VIEW SERVER STATE TO <login_name>;
GRANT VIEW ANY DEFINITION TO <login_name>;

USE <db>;
CREATE USER <user> FOR LOGIN <login_name>;
EXEC sp_addrolemember 'db_owner', '<user>';
EXEC sys.sp_cdc_enable_db
```

  For RDS for SQL Server:

```
USE master;
CREATE DATABASE <database>;
CREATE LOGIN <login> WITH PASSWORD = '<password>';
```

```
USE <database>;
EXEC msdb.dbo.rds_cdc_enable_db '<database>';
CREATE USER <user> FOR LOGIN <login_name>;

USE master;
GRANT VIEW SERVER STATE TO <login_name >;
GRANT VIEW ANY DEFINITION TO <login_name >;

USE <database>;
EXEC sp_addrolemember 'db_owner', '<user>';
```

- For database ingestion and replication incremental load and initial and incremental load jobs that have SQL Server sources and use log-based CDC, you must enable SQL Server Change Data Capture (CDC) on the source database.
  - For on-premises SQL Server sources, run the sys.sp_cdc_enable_db stored procedure in the database context. You must have the sysadmin role.
  - For Amazon Relational Database Service (RDS) for SQL Server sources, log in as the master user and run the msdb.dbo.rds_cdc_enable_db '*database_name*' stored procedure.

  After SQL Server CDC is enabled, SQL Server writes additional information to the transaction log and CDC tables, which Database Ingestion and Replication uses during incremental CDC processing.

  Alternatively, when you create a database ingestion and replication task, you have the option of generating a script that enables CDC on the database and on all columns in the selected source tables. To execute the CDC script, you need to have the sysadmin role.

  Restriction: Database Ingestion and Replication cannot enable CDC for tables that contain more than 1019 columns.
- For incremental load or combined initial and incremental load operations that use query-based CDC, the source table must contain the CDC query column that is used to indicate the changed rows. You must add the query column to the source tables before creating the database ingestion and replication task. The CDC query column type must be equivalent to timestamp, without a time zone. The supported SQL Server data types for the query column are DATETIME and DATETIME2.

Usage considerations

- Database ingestion and replication jobs support all load types for on-premises SQL Server, Amazon RDS for SQL Server, Azure SQL Managed Instance, and Azure SQL Database sources.
- Database Ingestion and Replication provides the following alternative capture methods for SQL Server sources in incremental load or combined initial and increment load jobs:
  - **CDC Tables**. Capture change data from the SQL Server CDC tables only.
  - **Log-based**. Captures change data from the SQL Server transaction log and CDC tables.

    Note: If you select this capture method for jobs that have Azure SQL Database sources, change data is read from CDC tables only, not from the transaction log.
  - **Query-based**. Capture Inserts and Updates by using a SQL WHERE clause that points to a CDC query column.

  The **CDC Tables** option provides the best replication performance and highest reliability of results.

  If you use the **Log-based** method for multiple jobs that run against the same database simultaneously, performance can be significantly degraded.

  For more information about each capture method, see "Change capture mechanisms for SQL Server sources" on page 33.
- When you enable CDC on the SQL Server database for a task that uses Log-based CDC, SQL Server automatically creates a capture job and a cleanup job that will be executed by the SQL Server Agent. The capture job is responsible for populating the SQL Server CDC tables. The cleanup job is responsible for cleaning up records from the CDC tables. The default value for data retention in the CDC table is 72 hours,

or 3 days. You can check the current retention period by running the sys.sp_cdc_help_jobs stored procedure and checking the retention value in the results. If you expect a downtime greater than 3 days, you can adjust the retention in the sys.sp_cdc_change_job stored procedure or in the SQL Server Agent cleanup job. You can also suspend the cleanup job.

- Database Ingestion and Replication supports SQL Server page compression and row compression of source data.
- Because Database Ingestion and Replication expects each source table row to be unique, Informatica recommends that each source table have a primary key. Database Ingestion and Replication does not honor unique indexes in place of a primary key. If no primary key is specified, Database Ingestion and Replication treats all columns as if they are part of the primary key. Exception: For SQL Server query-based CDC, a primary key is required in each source table.
- For log-based CDC with the transaction log, Database Ingestion and Replication requires read-write access on the source database. If you use SQL Server Always On availability groups, this requirement means that Database Ingestion and Replication can capture change data from the read-write primary replica but not from the read-only secondary replica.
- If a Microsoft SQL Server source uses the Always Encrypted method to encrypt column data, the CDC script that is generated from the **CDC Script** field on the **Source** page in the database ingestion and replication task fails to run. This problem is caused by a SQL Server limitation. This problem does not occur with Transparent Data Encryption (TDE).
- Database Ingestion and Replication supports schema drift options for Microsoft SQL Server sources in database ingestion and repoliclationincremental load jobs. The following limitations apply:
  - Microsoft SQL Server does not support renaming tables and columns for which Change Data Capture (CDC) is enabled.
  - Microsoft SQL Server does not support changing primary keys for CDC tables.
  - When Database Ingestion and Replication reads change data directly from the CDC tables, the CDC tables do not change once they are created. DDL changes that occur on the source tables are replicated as nulls in the CDC tables. To replicate the DDL changes to the CDC tables, set the pwx.custom.sslr_cdc_manage_instances custom property to 1 on the **Source** page of the task wizard. This custom property enables the modification of the CDC tables to reflect DDL changes on the source tables and enhance DML capture. You must have the db_owner role to enable the active management of the CDC tables.
- If source table partition changes cause rowset IDs to change, Database Ingestion and Replication can process the changes to enable database ingestion and replication jobs to continue capturing DML changes from the tables.
- For incremental load and initial and incremental load jobs that use the query-based CDC method, the following limitations apply:
  - A primary key is required in each selected source table. If a primary key is not present in a source table, change data capture ignores the table and continues processing with the rest of the selected source tables. If none of the source tables have a primary key, the job will fail.
  - Query-based CDC does not capture Delete operations.
  - All Insert and Update operations are treated as Upserts and displayed in the monitoring interface and logs as Updates.
  - Data replication from large-object (LOB) columns is not supported. If the source table contains LOB columns, Database Ingestion and Replication propagates nulls for these columns.
  - If a Daylight Savings Time change or a time zone change is detected at the start of a particular cycle or when the job resumes from a failed or stopped state, Database Ingestion and Replication will resume and process the changes that occurred in that cycle.

- Database ingestion and replication jobs can replicate data from Microsoft SQL Server large-object (LOB) columns if you select **Include LOBs** under **Advanced** on the **Source** page of the task wizard. The supported target types depend on the load type:
  - For initial load jobs: Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, Microsoft Azure Synapse Analytics, Oracle, Oracle Cloud Object Storage, Snowflake, and SQL Server.
  - For incremental load and combined initial and incremental load jobs: Azure Event Hubs, Databricks, Snowflake, and SQL Server.

  LOB data types are GEOGRAPHY, GEOMETRY, IMAGE, VARBINARY(MAX), VARCHAR(MAX), NVARCHAR(MAX), TEXT, NTEXT, and XML. LOB data might be truncated before being written to the target. The truncation point, depends on the data type, target type, and load type. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.
- Database Ingestion and Replication does not replicate data from SQL Server non-persisted computed columns. For initial load jobs and for incremental load and combined initial and incremental load jobs that use the log-based or query-based CDC method, persisted computed columns are replicated to the target. For incremental load and combined initial and incremental load jobs that capture changes from CDC Tables only, persisted computed columns are replicated as NULLs or as an empty value, depending on column nullability.
- Database ingestion and replication initial load jobs that have a SQL Server source and a SQL Server target and include a sql_variant source column convert the sql_variant data to hexadecimal format on the target. To convert data from hexadecimal format to varbinary format, run the following query:

  ```
  SELECT <column_name>, CONVERT(varbinary,<column_name>) from <table_name>;
  ```

  Replace <*column_name*> and <*table_name*> with the actual target column and table names.
- The SQL Server Hierarchyid data type is not supported in database ingestion and replication incremental load and combined initial and incremental load jobs that have a Snowflake target. Database Ingestion and Replication will propagate nulls for columns that have this data type. For more information, see "Default Data Type Mappings" on page 172.
- If you use a Secure Agent group with multiple agents and the active agent goes down unexpectedly, database ingestion and replication initial load, incremental load, and combined initial and incremental load jobs can automatically switch over to another available agent in the Secure Agent group. The automatic switchover occurs after the 15 minute heartbeat interval elapses. Automatic switchovers are supported for initial load jobs with any source type and a target type other than Kafka. They're also supported for incremental load and combined load jobs that have a SQL Server source, subject to following limitations:
  - The job cannot have a Kafka target.
  - The job cannot have persistent storage enabled.
  - The job cannot use the Query-based CDC method to capture changes from the SQL Server source.
- If a SQL Server source is in an Always On availability group, database ingestion and replication incremental load and combined load jobs can capture change data from transaction logs or CDC tables on the primary node or a secondary node. Also, if a node becomes unavailable, database ingestion and replication jobs can fail over to a primary or secondary database in an availability replica to continue processing, provided that you've configured the SQL Server connection to point to an availability group listener. An availability group listener is a virtual network name (VNN) that Database Ingestion and Replication can use to access a database in an availability replica of an availability group, without having to know the SQL Server physical instance name.
- After a database ingestion and replication job with an SQL Server source has run, if you select additional source columns for replication and redeploy the task, the job does not immediately re-create the target table with the additional columns or replicate data for them. However, an incremental load or combined

initial and incremental load job will add the newly selected columns to the target and replicate data to them when it processes the next new DML change record, provided that you set the schema drift **Add Column** option to **Replicate**. An initial load job will add the newly selected columns to the target and replicate data to them the next time the job runs.

- SQL Server memory-optimized tables are supported as sources in database ingestion and replication incremental load and combined initial and incremental load jobs that use the **Query-based** CDC method. However, memory-optimized tables are not supported as sources in jobs that use the **CDC Tables** or **Log-based** CDC method.
- If your SQL Server sources include persisted computed columns, database ingestion and replication incremental load and combined load jobs can replicate the computed column expression values, regardless of which CDC method you use.

Change capture mechanisms for SQL Server sources

Database Ingestion and Replication provides multiple change capture mechanisms for capturing change data from a SQL Server source and applying the data to a target.

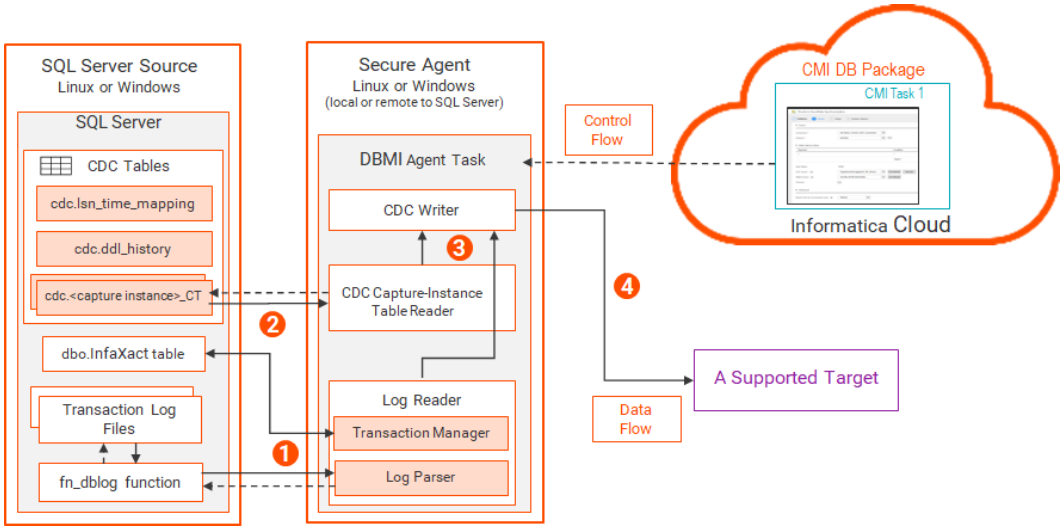
Log-based change data capture with transaction log and CDC tables

Database ingestion and replication incremental load jobs that use log-based CDC capture DML and DDL changes by parsing records from the active transaction log and by reading change records directly from the CDC tables. Change data is read from the active transaction log if the required restart point (LSN) is available there. If the capture start point pre-dates records in the transaction log or in certain other situations, Database Ingestion and Replication automatically transitions to reading change data from the CDC tables. After reading changes from the CDC tables, Database Ingestion and Replication switches back to reading changes from the active transaction log in a transparent manner.

The following components are involved in log-based change capture processing:

- The Informatica Intelligent Cloud Services Secure Agent, which can run on a Linux or Windows box that is local to or remote from the SQL Server instance.
- The dbo.$InfaXact table, which is associated with an open transaction and temporarily stores some information about the currently running capture job. If the table does not exist, the change capture process creates it.
- The Log Reader and its subcomponents, which are required to parse and read DML and DDL change records from the transaction log.
- The SQL Server fn_dblog() function, to which the Log Reader loops calls to retrieve log records from the active part of the transaction log file for the source database.
- The SQL Server CDC tables: cdc.lsn_time_mapping, cdc.ddl_history, and cdc.<*capture instance*>_CT. SQL Server creates these tables in the CDC schema when the source database is enabled for CDC. One or two cdc.<*capture instance*>_CT tables are created for each CDC-enabled source table to store DML changes captured by the native log reader.
- The CDC Capture-Instance Table Reader, which reads change records from the cdc.<*capture_instance*>_CT tables.
- The CDC Writer, which writes the changes to the target.

The following image shows the SQL Server log-based change data capture components and data flow:



1. The Log Reader process reads log records to capture DML and DDL changes in committed transactions.
   - The Transaction Manager subcomponent interacts with the dbo.$InfaXact table associated with the open transaction.
   - The Log Parser subcomponent loops calls to the fn_dblog() function to read log records from the active transaction log for the selected source tables that are enabled for CDC.

   Capture processing continues until the committed transactions are complete, the capture process is stopped or interrupted by a fatal error, or a switch to reading change records from a cdc.<*capture_instance*>_CT table is triggered.
2. In certain situations, Database Ingestion and Replication automatically switches to reading changes from a cdc. <*capture instance*>_CT table by using the CDC Capture-Instance Table Reader. Processing switches to the cdc.<*capture instance*>_CT table under the following conditions:
   - During initialization and startup of the capture process if the start LSN is earlier than the LSN of the active transaction log and exists within the cdc.<*capture instance*>_CT table.
   - Transaction log truncation occurs, causing data loss. Normally, SQL Server prevents log truncation when an open transaction is associated with dbo.$InfaXact . However, a lost network connection might end a transaction, causing log truncation.

     Note: Routine log backups can also truncate the transaction log. To prevent data loss, transactions that use the dbo.$InfaXact table lock the active transaction log.
   - Off-row LOBs or primary row records are truncated. In this situation, the Log Reader selectively patches the log records by reading column information from the cdc.<*capture instance*>_CT tables.

     Note: Records read from fn_dblog() function are truncated to 8000 bytes.
3. The Log Reader and CDC Capture-Instance Table Reader send change records to the CDC Writer.
4. The CDC Writer formats the change records and applies them to the target.

Change capture with CDC tables only

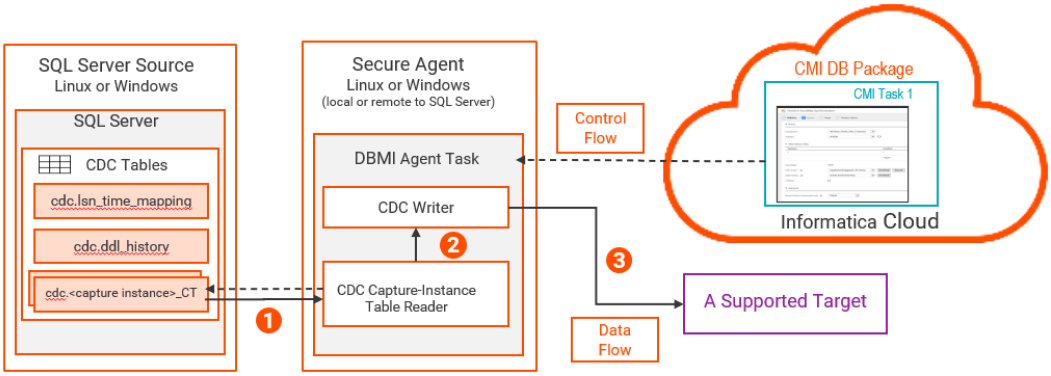
Database ingestion and replication incremental load jobs can capture changes directly from the SQL Server CDC tables without using the transaction log.

The following components are involved in the change capture processing that uses CDC tables only:

- The Informatica Intelligent Cloud Services Secure Agent, which can run on a Linux or Windows box that is local to or remote from the SQL Server instance.

- The SQL Server CDC tables: cdc.lsn_time_mapping, cdc.ddl_history, and cdc.<*capture instance*>_CT. SQL Server creates these tables in the CDC schema when the source database is enabled for CDC. One or two cdc.<*capture instance*>_CT tables are created for each CDC-enabled source table to store DML changes captured by the native log reader.
- The CDC Capture-Instance Table Reader, which reads change records from the cdc.<*capture_instance*>_CT tables.
- The CDC Writer, which writes the changes to the target.

The following image shows the SQL Server log-based change data capture components and data flow:



1. The CDC Capture-Instance Table Reader reads changes from a cdc. <*capture instance*>_CT table.
2. The CDC Capture-Instance Table Reader sends change records to the CDC Writer.
3. The CDC Writer formats the change records and applies them to the target.

Query-based change capture

Database ingestion and replication jobs capture insert and update changes in the source by querying a timestamp column that is updated when a change occurs. Configuration of the source is limited to adding the common CDC query column to each source table. The query-based CDC method uses the query column to identify the rows that changed since the beginning of a specified CDC interval.

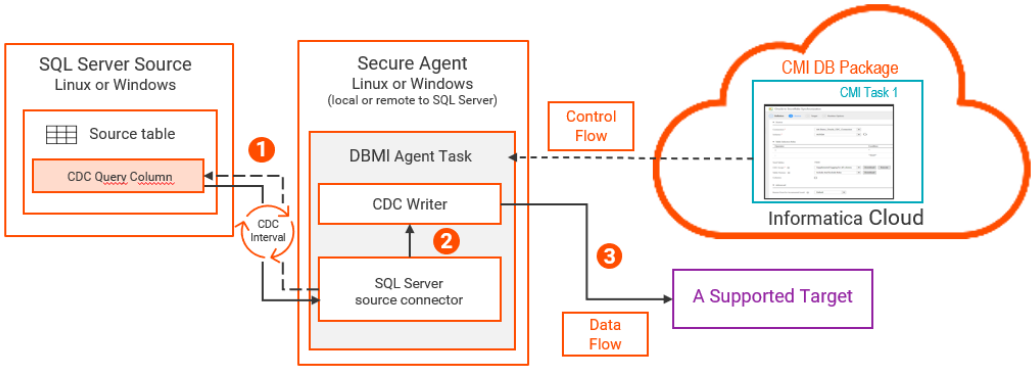
To implement the query-based change capture, set the following options on the **Source** page of the task wizard:

- **CDC Method**. Select **Query-based** to enable this capture method.
- **CDC Query Column Name**. The case-sensitive name of the CDC query column in the source table. The supported SQL Server data types for the query column are DATETIME and DATETIME2. The column must be present in the source table.
- **CDC Interval**. The frequency of a query-based change data capture cycle. Default is 5 minutes.
- **Initial Start Point for Incremental Load**. The point from which you want to start the change capture cycle. Default is **Latest Available**.

After the CDC interval elapses, Database Ingestion and Replication uses a SQL query with a WHERE clause that references the CDC query column to identify the rows that received changes during the CDC interval. The change data is captured and applied to the target.

If the source tables selected for the query-based CDC do not have the CDC query column, change data capture will ignore these tables and will continue with the rest of the tables. For the tables that are skipped, corresponding tables generated in the target database will be empty. If none of the source tables have a CDC query column, the job will fail at runtime.

The following image shows the SQL Server query-based change data capture components and data flow:



1. After the CDC interval elapses, Database Ingestion and Replication executes a SQL query in the source database that uses the CDC query column to extract the change data.
2. Change records are sent to the CDC Writer.
3. The CDC Writer formats the change records and applies them to the target.

MongoDB sources

To use MongoDB sources in database ingestion and replication tasks, review the following considerations.

Source preparation

- Database Ingestion and Replication uses MongoDB change streams to access real-time data changes on a single collection, a database, or an entire deployment.
- For incremental load jobs, ensure that the database user you specify in the MongoDB connection has the readWriteAnyDatabase role and the privileges that grant the changeStream and find actions.
- To open a change stream on a single database, applications must have privileges that grant changeStream and find actions on all non-system collections in the database. Use the following statements to grant these privileges:

```
{ resource:
{ db: <dbname>, collection: "" }
, actions: [ "find", "changeStream" ] }
```

  Example:

```
db.createRole({ role: "readWriteAnyDatabase", privileges: [{ resource:
{ db: "<Database>", collection: "" }
, actions: [ "find", "changeStream" ] }], roles: []})
```

- To open a change stream on an entire deployment, applications must have privileges that grant changeStream and find actions on all non-system collections for all databases in the deployment. Use the following statements to grant these privileges:

```
{ resource:
{ db: "", collection: "" }
, actions: [ "find", "changeStream" ] }
```

Usage considerations

- The database ingestion and replication task moves the MongoDB data to the target as key-value pairs, where the *key* is the ObjectID and the *value* is the JSON string that comprises a BSON document.
- For MongoDB sources, data type mappings do not occur. All data is persisted on the target as string data.
- In incremental load operations, the data change at the source is tracked by means of a unique key (ObjectID) and the same changed JSON string is applied to the target.

- Database Ingestion and Replication does not support time series collections in incremental load jobs that have MongoDB sources.
- In incremental load operations, Database Ingestion and Replication retrieves the change records from the date and time specified as the restart point. For MongoDB sources, the default value for the restart point is the current time. You can specify a different date and time in Greenwich Mean Time (GMT).
- If schema drift occurs on the MongoDB source, the data in BSON documents that are sent to the target reflect the schema changes. However, Database Ingestion and Replication does not specifically detect and report the schema changes.

MySQL sources

To use MySQL sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations.

Source preparation

- To deploy and run a database ingestion and replication task that includes a MySQL source, the source connection must specify a database user who has the required privileges. Use the following SQL statements to grant these privileges to the user:

  ```
  GRANT SELECT ON database_name.* TO 'user_name'@'%';
  GRANT SELECT ON TABLE database_name.* TO 'user_name'@'%';
  ```

  For incremental load jobs, grant the following additional privileges to the user:

  ```
  /* To fetch table and column details from system tables */
  GRANT SELECT ON sys.* TO 'user_name'@'%';

  /* To allow the user to monitor binary log information such as file name, position,
  and GTID */
  GRANT REPLICATION CLIENT ON *.* TO 'user_name;

  /* Required for a multi-node MySQL cluster with master and slave configuration */
  GRANT REPLICATION SLAVE ON *.* TO 'user_name;
  ```

- For incremental load jobs, set the default_storage_engine variable to InnoDB in the [mysqld] section of the my.cnf file. Then restart the MySQL server. To verify the default_storage_engine setting, use the following statement:

  ```
  SHOW VARIABLES LIKE '%engine%';
  ```

  In the output, make sure that the default_storage_engine variable is set to InnoDB.

- Database Ingestion and Replication uses binary log files generated by MySQL for capturing change events at the source. The binlog is a set of log files that contain information about data modifications made to a MySQL server instance.
  To enable binary logging, start the server with the `--log-bin` option or use the key-value `log-bin="[HostName]-bin"` setting in the my.cnf file. Replace `[HostName]` with the name of your host. Then restart the MySQL server. To verify that binary logging is enabled, use the following statement:

  ```
  SHOW VARIABLES LIKE 'log_bin';
  ```

  In the output, make sure that the log_bin variable is set to ON.

- For incremental load jobs, enable row-based logging by using the following statement:

  ```
  SET GLOBAL binlog_format = 'ROW';
  ```

  To verify that row-based logging is enabled, use the following statement:

  ```
  SHOW VARIABLES LIKE 'binlog_format';
  ```

  In the output, make sure that the binlog_format system variable is set to ROW.

- To enable database ingestion and replication incremental load and combined load jobs to process Updates and Deletes, set the following system variable for binary logging:

  ```
  binlog_row_image=full
  ```

  This setting causes both before images and after images to be logged to the binary log for all columns. It's applicable to any MySQL source type and version that Database Ingestion and Replication supports.

  To verify this setting, use the following statement:

  ```
  SHOW VARIABLES LIKE 'binlog_row_image';
  ```

- Database Ingestion and Replication can read the binlog files in either one of the following ways:

  - Global Transaction ID (GTID) - If you enable MySQL GTID mode, every transaction in MySQL is assigned a GTID to uniquely identify the transaction. Use GTID mode in a multi-cluster environment.
  - Binlog file name and position - All transactions in MySQL are saved as anonymous and fetched by using the binlog file name and position. Do not use this method if the MySQL GTID mode is enabled or if you have a multi-cluster environment. In a multi-cluster environment, the binlog file position might vary if a failover occurs, causing inconsistent data.

  To enable the GTID mode, use the following statements on each MySQL server:

  ```
  SET @@GLOBAL.ENFORCE_GTID_CONSISTENCY = WARN;
  SET @@GLOBAL.ENFORCE_GTID_CONSISTENCY = ON;
  SET @@GLOBAL.GTID_MODE = OFF_PERMISSIVE;
  SET @@GLOBAL.GTID_MODE = ON_PERMISSIVE;
  ```

  On each server, wait until the status variable 'Ongoing_anonymous_transaction_count' is 0 (zero). You can check the status variable's value by using the following statement:

  ```
  SHOW STATUS LIKE 'Ongoing_anonymous_transaction_count';
  ```

  When the count is 0, use the following statement to enable GTID mode:

  ```
  SET @@GLOBAL.GTID_MODE = ON;
  ```

- You must download the MySQL driver file and copy it to a specific installation subdirectory to be able to connect to a MySQL source, if you use the following load type and source edition combinations:

  - Incremental load jobs that have MySQL Community Edition or MySQL Enterprise Edition sources
  - Initial load jobs that have MySQL Community Edition sources
  - Initial load jobs that have Amazon Relational Database Service (RDS) for MySQL sources

  Note: You do not need to download the driver if you only run initial load jobs that have MySQL Enterprise Edition sources.

  Download the MySQL JDBC driver file, mysql-connector-java-<version>.jar, from the MySQL Community Downloads website and copy it to the following directory:

  ```
  <Secure_Agent_installation_directory>/ext/connectors/thirdparty/com.mysql/
  ```

  If you want to be able to test a connection in Administrator after defining connection properties, you must also set the MySQL_JDBC_DRIVER_JARNAME parameter for the Data Integration Server service in the Secure Agent's system configuration details. After the test, you can remove the parameter. This parameter is not used when you use the connection to create a database ingestion and replication task or run the associated job.

Usage considerations

- Database Ingestion and Replication supports MySQL, Amazon Aurora MySQL, Cloud SQL for MySQL, and RDS for MySQL sources for initial load, incremental load, and combined initial and incremental load jobs. Also, Database Ingestion and Replication supports Azure Database for MySQL sources with Snowflake targets for all load types and with Confluent Kafka for incremental loads.
- Because Database Ingestion and Replication expects each source table row to be unique, Informatica recommends that each source table have a primary key. Database Ingestion and Replication does not

honor unique indexes in place of a primary key. If no primary key is specified, Database Ingestion and Replication treats all columns as if they are part of the primary key.

- If you update the primary key value in the MySQL source table for a record that does not exist in the target table, the record is not replicated to the target. However, the monitoring interface increments the Update count to include the primary key update. The data will replicate to the target if the record already exists in the target table before the update of the primary key value.
- Database ingestion and replication jobs that contain a MySQL source column with the SET or ENUM data type replicate the SET and ENUM column data as numeric values to the target. For initial load jobs, you can set the mysql.set.and.enum.as.numeric custom property to false on the **Source** page of the task wizard to replicate the SET or ENUM data in string or varchar format. The default value is true, which causes Database Ingestion and Replication to replicate the SET or ENUM data as numeric values.

  Note: For incremental load jobs, you cannot toggle between the numeric and string or varchar representation of the SET or ENUM column data. If you set the mysql.set.and.enum.as.numeric custom property to false and run an initial load job, followed by an incremental load job, Database Ingestion and Replication replicates the SET and ENUM data to the target as numeric values only.
- Database Ingestion and Replication does not support the following MySQL data types:

  - BLOB
  - JSON
  - LONGBLOB
  - LONGTEXT
  - MEDIUMBLOB
  - MEDIUMTEXT
  - TEXT
  - TINYBLOB
  - TINYTEXT

  If a database ingestion and replication task specifies a source schema that includes columns with the JSON data type, deployment of the task ignores the JSON columns and does not create corresponding columns on the target. For other unsupported data types, database ingestion jobs propagate nulls.

  For information about the default mappings of supported source data types to target data types, see "Default Data Type Mappings" on page 172.

Netezza sources

To use Netezza sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations.

Source preparation

- Download and install the Netezza JDBC driver.
  1. Download the Netezza JDBC driver from the IBM website.
  2. Copy the Netezza JDBC driver jar file, nzjdbc.jar, to the following directory:

     ```
     <Secure Agent installation directory>/apps/Database_Ingestion/ext/
     ```
  3. Restart the Secure Agent.
- To deploy and run a database ingestion and replication task that includes a Netezza source, the source connection must specify a database user who has the privileges that are required to perform an initial

load operation. Configure SELECT permissions for the Netezza user account on the following system views:

- _V_JDBC_SCHEMA1
- _V_JDBC_SCHEMA3
- _V_ODBC_TABLES3
- _V_ODBC_COLUMNS3
- _V_ODBC_PRIMARYKEYS3

Usage considerations

- Database Ingestion and Replication does not support the following Netezza data type:

  - ST_GEOMETRY

  Database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have this data type.

  For information about the default mappings of supported source data types to target data types, see "Default Data Type Mappings" on page 172.

Oracle sources

To use Oracle sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations.

Source preparation

- Define the following system environment variables on the Linux or Windows system where the Secure Agent runs:

  - ORACLE_HOME environment variable. Points to the Oracle client installation directory, one level up from the bin directory on Windows or lib directory on Linux. This environment variable is not required. However, if you do not define it, you'll need to specify the full Oracle client installation path when you define other environment variables or Secure Agent properties that specify a path to a subdirectory.
  - TNS_ADMIN environment variable. If you specify a TNS name in the **Database Connect String** property of the Oracle Database Ingestion connection properties, use this environment variable to point to the directory location of the tsnnames.ora file when the file is not in the default $ORACLE_HOME/network/admin directory. The tnsnames.ora file, along with the Oracle Call Interface (OCI), is used to communicate with the Oracle source database.

  Note: In Administrator, you can set the ociPath property for the Database Ingestion agent service (DBMI agent) to point to the OCI library that contains the oci.dll or libclntsh.so file. The OCI library is used by database ingestion and replication CDC tasks to connect to Oracle. By default, Oracle uses the ociPath value of $ORACLE_HOME/lib on Linux or %ORACLE_HOME%\bin on Windows.

- Make sure that the Database Ingestion and Replication user has the Oracle privileges that are required for the database ingestion and replication load type to be performed.

  Note: For combined initial and incremental loads that use log-based CDC, make sure that the GRANT FLASHBACK privilege is issued for each selected source table or use the ANY TABLE option. Database Ingestion and Replication uses an Oracle Flashback Query, which consists of a SELECT AS OF *scn* statement, to query for row data from source tables in the Oracle database. Oracle requires the GRANT FLASHBACK privilege for this query to be used.

  For more information, see "Oracle privileges" on page 48.

- If you use Oracle 11.2.04, set the Oracle COMPATIBLE initialization parameter to 11.2.04 to ensure that Oracle has all of the most current Redo Logs fixes for that release.

- Database ingestion and replication jobs that use log-based CDC require read access to Oracle online and archive redo logs to read incremental change data. If the redo logs are remote from the on-premises system where the Secure Agent runs, make sure that read access to the logs is provided, for example, by using Oracle Automatic Storage Management (ASM), mounting the logs to a network file system (NFS), or configuring BFILE access to logs that are on the Oracle file system.
- If you plan to read data from redo log files in Oracle ASM, Informatica recommends that you set the sqlnet.recv_timeout parameter in the local sqlnet.ora file to less than 5 minutes. This parameter specifies the duration of time that the Oracle client waits for a response from ASM before a query times out. Network interrupts and other factors can occasionally make Oracle connections unresponsive. Setting this value ensures that the reader can respond and recover from any such situation in a timely manner.
- Ensure that the Oracle Database Client or Instant Client is installed and configured on the Secure Agent server for the Secure Agent to communicate with Oracle. If you do not already have an Oracle client installed, you can download a client and access installation information from the Oracle web site, or ask your Oracle DBA to download and configure an Oracle client.
- For incremental load or combined initial and incremental load operations that use log-based CDC, perform the following prerequisite tasks in Oracle:
  - Enable ARCHIVELOG mode for the Oracle database. If the database is not in an Amazon RDS environment, issue the following SQL statements:

    ```
    SHUTDOWN IMMEDIATE;
    STARTUP MOUNT;
    ALTER DATABASE ARCHIVELOG;
    ALTER DATABASE OPEN;
    SHUTDOWN IMMEDIATE;
    STARTUP;
    ```

    For an Amazon RDS for Oracle databases, set the backup retention period to place the database in ARCHIVELOG mode and enable automated backups.
  - Define an archive log destination.
  - Enable Oracle minimal global supplemental logging on the source database.
  - If your Oracle source tables have primary keys, ensure that supplemental logging is enabled for all primary key columns. For source tables that do not have primary keys, ensure that supplemental logging is enabled for all columns from which change data will be captured.

    Note: When you create a database ingestion and replication task, you have the option of generating a script that implements supplemental logging for all columns or only primary key columns for the selected source tables.
  - Ensure that the Oracle MAX_STRING_SIZE initialization parameter is *not* set to EXTENDED. If it is set to EXTENDED, Database Ingestion and Replication will not be able to replicate inserts and updates for tables containing columns defined with large (extended size) VARCHAR2, NVARCHAR2, or RAW columns.

  If you do not have the authority to perform these tasks, ask your Oracle database administrator to perform them. For more information, see the Oracle documentation.
- For incremental load operations that use query-based CDC, the source table must contain the CDC query column that is used to indicate the changed rows. You must add the query column to the source tables before creating the database ingestion and replication task. The supported Oracle data type for the query column is TIMESTAMP.
  If the source tables selected for the query-based CDC do not have the CDC query column, change data capture will ignore these tables and will continue with the rest of the tables. For the tables that are skipped, corresponding tables generated in the target database will be empty. If none of the source tables have a CDC query column, the deployment of the job will fail.

Amazon Relational Database Service (RDS) for Oracle source preparation:

1. Create the ONLINELOG_DIR and ARCHIVELOG_DIR directories that will hold the online and archive redo logs, respectively, on the RDS file system. Use the following exec statements:

```
exec rdsadmin.rdsadmin_master_util.create_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
```

2. Grant the Oracle privileges that are required for the Amazon RDS for Oracle source type to the Database Ingestion and Replication user.
   For more information about the privileges required for an Amazon RDS for Oracle source, see "Oracle privileges for Amazon RDS for Oracle sources" on page 50.

3. Define an appropriate retention time for the archived redo logs. Use the following exec statement:

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention
days',number_of_days);
```

4. In the Amazon RDS console, set the backup retention period for the source database to a value greater than zero to enable automated backups of the database instance.

   Note: This step enables ARCHIVELOG mode for the database.

5. Ensure that supplemental logging is enabled at the database level. Use the following statement:

```
exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD');
```

   When you create a database ingestion and replication task, you can generate a script to enable supplemental logging for the selected source tables.

6. Optionally, in the Amazon RDS console, you can create a parameter group and define the cache sizes of the default buffer pool. The default buffer pool holds buffers that use the primary database block size. Use the following DB_CACHE_SIZE parameter values:

   - DB_2K_CACHE_SIZE
   - DB_4K_CACHE_SIZE
   - DB_16K_CACHE_SIZE
   - DB_32K_CACHE_SIZE

   Then select the parameter group for the source database.

Usage considerations

- For incremental load operations with Oracle sources, Database Ingestion and Replication provides alternative capture methods for capturing change data from the source and applying the data to a target. The available change capture methods are:
  - Log-based change data capture. Database Ingestion and Replication reads data changes from Oracle redo logs. This method requires users to have extended privileges.
  - Query-based change data capture. Change data capture uses a SQL statement with a WHERE clause that references a common CDC query column to identify the rows with insert and update changes. Configuration of the source database is limited to adding the CDC query column to each source table. Users must have at least read only access to the source tables.
- Because Database Ingestion and Replication expects each source table row to be unique, Informatica recommends that each source table have a primary key. Database Ingestion and Replication does not honor unique indexes in place of a primary key. If no primary key is specified, Database Ingestion and Replication treats all columns as if they are part of the primary key. Exception: For Oracle query-based CDC, a primary key is required in each source table.
- For Oracle sources that use the multitenant architecture, the source tables must reside in a single pluggable database (PDB) within a multitenant container database (CDB).
- You can use Oracle Transparent Data Encryption (TDE) to encrypt data in tablespaces that contain Oracle source tables for incremental load processing. Database Ingestion and Replication supports storing the

master encryption key in a TDE keystore that is in a file system, in ASM, or in an external hardware security module (HSM) that supply PKCS11 interfaces, such as Oracle Key Vault (OKV). For more information, contact Informatica Global Customer Support.

- If Oracle source CHAR or VARCHAR columns contain nulls, the database ingestion and replication job does not delimit the null values with double-quotation (") marks or any other delimiter when writing data to a Amazon S3, Flat File, Microsoft Azure Data Lake, or Microsoft Azure Synapse Analytics target.
- Database Ingestion and Replication supports Oracle Data Guard logical and physical standby databases and far sync instances as sources. For more information, see "Oracle Data Guard databases or far sync instances as sources" on page 58.
- Database Ingestion and Replication can process data across a RESETLOGS boundary. To avoid the source and targets becoming out of sync, Informatica recommends that you stop capture processing before performing a RESETLOGS and then restart capture processing after the RESETLOGS event. Otherwise, the capture process might send data to the target that is subsequently reverted by the RESETLOGS event, causing the source and target to become out of sync.
- Alternative strategies for accessing the Oracle redo logs are available. For more information, see "Oracle log access methods for CDC" on page 53.
- If a database ingestion and replication incremental load or combined initial and incremental load task contains an Oracle source table name or one or more column names that are longer than 30 characters, Oracle suppresses supplemental logging for the entire table, including primary keys and foreign keys. As a result, most operations on the table fail. This problem is caused by an Oracle restriction. In this situation, exclude the table from capture processing or rename the long table and column names to names of 30 characters or less.
- Database Ingestion and Replication can replicate data from Oracle BLOB, CLOB, NCLOB, LONG, LONG RAW, and XML columns to Amazon Redshift, Amazon S3, Databricks, Google Big Query, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, Microsoft Azure Synapse Analytics, Microsoft Fabric OneLake, Oracle, Oracle Cloud Object Storage, PostgreSQL, Snowflake, and SQL Server targets. You must select **Include LOBs** under **Advanced** on the **Source** page when you configure the task. LOB column data might truncated on the target if it is greater in size than the byte limit allowed by target type.

  Note: Columns that have the LONG, LONG RAW, and XML data types are supported in initial load jobs and in incremental load and combined load jobs that use the Query-based CDC method. However, jobs that use the Log-based CDC method do not replicate data from these types of columns to the generated target table.

  For more information, see "Configuring the source" on page 106.
- Database Ingestion and Replication does not support the following Oracle source data types with any target type or load type:

  - "ANY" types such as ANYTYPE, ANYDATA, ANYDATASET
  - Extended types
  - INTERVAL
  - JSON
  - BFILE type for storing LOB data externally
  - UROWID
  - Spatial types such as SDO_GEOMETRY
  - User-defined types such as OBJECT, REF, VARRAY, nested table types

  Source columns that have unsupported data types are excluded from the target definition.

  For information about the default mappings of supported Oracle data types to target data types, see "Default Data Type Mappings" on page 172.

- To use the Oracle TIMESTAMP WITH LOCAL TIME ZONE data type, set the DBMI_ORACLE_SOURCE_ENABLE_TIMESTAMP_WITH_LOCAL_TZ environment variable to true for the Database Ingestion agent service. In Administrator, open your Secure Agent and click **Edit** . Under **Custom Configuration Details**, add the environment variable with the following details:
  - **Service**: Database Ingestion
  - **Type**: DBMI_AGENT_ENV
  - **Name**: DBMI_ORACLE_SOURCE_ENABLE_TIMESTAMP_WITH_LOCAL_TZ
  - **Value**: true
- Do not use custom data type mapping rules to map an Oracle source RAW column to a target CHAR or VARCHAR column. Otherwise, the deployment of the database ingestion and replication task might fail.
- Database Ingestion and Replication does not support invisible columns in Oracle source columns, regardless of the target type. For these columns, database ingestion and replication incremental load jobs and combined initial and incremental load jobs propagate nulls to the corresponding target columns.
- If you update the primary key value in the Oracle source table for a record that does not exist in the target table, the record is not replicated to the target. However, the monitoring interface increments the Update count to include the primary key update. The data will replicate to the target if the record already exists in the target table before the update of the primary key value.
- If an Update to an Oracle table does not change an existing column value, the Update count in the monitoring details for the table is still incremented but no Update row is applied to the target. Database Ingestion and Replication ignores Update rows that do not actually change values. Also, for most database targets, Database Ingestion and Replication does some aggregation of change records at a micro-batch level before writing changes to the targets. This situation can also lead to a mismatch between the Update count in the monitoring statistics and the rows applied to the target.
- The supplemental logging settings for tables might be ignored by Oracle if the table name or any table column name is longer than 30 characters. In this case, the results of database ingestion and replication incremental load or combined load jobs are unpredictable.
- Database Ingestion and Replication does not support derived columns in jobs that have an Oracle source.
- For Oracle combined initial and incremental load jobs, Oracle Flashback queries are used to get committed data that was current at a specific point in the change stream. Ensure that no source table is truncated during the initial load period. If truncation occurs, any DDL change performed during a flashback query causes the query to fail.
- If you use a Secure Agent group with multiple agents and the active agent goes down unexpectedly, database ingestion and replication initial load, incremental load, and combined initial and incremental load jobs can automatically switch over to another available agent in the Secure Agent group. The automatic switchover occurs after the 15 minute heartbeat interval elapses. Automatic switchovers are supported for initial load jobs with any source type and a target type other than Kafka. They're also supported for incremental load and combined load jobs that have an Oracle source, subject to following limitations:
  - The job cannot have a Kafka target.
  - The job cannot have persistent storage enabled.
  - The job cannot use the Query-based CDC method to capture changes from the Oracle source.
- For incremental load and combined initial and incremental load jobs that use the query-based CDC method, the following limitations apply:
  - A primary key is required in each selected source table. If a primary key is not present in a source table, change data capture ignores the table and continues processing with the rest of the selected source tables. If none of the source tables have a primary key, the job will fail.

- Query-based CDC does not capture Delete operations.
- All Insert and Update operations are treated as Upserts and displayed in the monitoring interface and logs as Updates.
- If a Daylight Savings Time change or a time zone change is detected at the start of a particular cycle or when the job resumes from a failed or stopped state, Database Ingestion and Replication will resume and process the changes that occurred in that cycle. You must restart the Oracle database to apply a Daylight Savings Time change or a time zone change.
- Database ingestion and replication combined initial and incremental load jobs that have Oracle sources can read changes from copies of the archive redo logs. You must set the **Reader Mode** property to ARCHIVECOPY in the Oracle Database Ingestion connection properties and also set the source custom property pwx.cdcreader.oracle.reader.additional with the dir and file parameters. The dir parameter points to the name of the base directory that the CDC log reader scans for the archive log copies, and the file parameter specifies a mask that is used to filter the log copies.

- After a database ingestion and replication job with an Oracle source has run, if you select additional source columns for replication and redeploy the task, the job does not immediately re-create the target table with the additional columns or replicate data for them. However, an incremental load or combined initial and incremental load job will add the newly selected columns to the target and replicate data to them when it processes the next new DML change record, provided that you set the schema drift **Add Column** option to **Replicate**. An initial load job will add the newly selected columns to the target and replicate data to them the next time the job runs.
- Database Ingestion and Replication can capture change data from Oracle Exadata machines but does not support Oracle Exadata Hybrid Columnar Compression (EHCC).
- Database Ingestion and Replication jobs of any load type cannot use Oracle synonyms as sources.
- If LOB data is incompletely logged for an Oracle source for any reason, database ingestion and replication jobs that process the source and have the **Include LOBs** option enabled will issue a warning message such as:

  ```
  PWX-36678 ORAD WARN: Failed to fetch LOB data for table <schema>.<table> column
  <column> to resolve partially logged data.
  ```

  Usually, data is incompletely logged because it was appended to an existing LOB column. This issue can occur for any load type.

  To resolve the issue, specify the pwx.cdcreader.oracle.option.additional custom property with the LOB_FETCHBACK=Y parameter on the **Source** page of the task wizard. This setting causes the current data to be fetched directly from the database, instead of from the logs. To successfully use this solution, the user must be granted the SELECT ON ANY TABLE privilege.

Gathering Information About the Database Ingestion and Replication environment

Before you start creating database ingestion and replication tasks, gather the following information:

General Information

What Oracle version do you use?

Answer: _____

Do you run Oracle on premises or in a cloud-based Amazon RDS for Oracle environment?

Answer: _____

What is the target type?

Answer: _____

What type of load operation do you plan to perform: an initial load (point-in-time bulk load), incremental load (only the changes), or combined initial and incremental load (initial load followed by incremental load)?

Answer: _____

What are the number of cores, memory, and disk space on the system where the Secure Agent will run?

Answer: _____

Oracle environment

What are the host name and port number of the Oracle source database server?

Answer: _____

What is the Oracle system identifier (SID) for the database?

Answer: _____

What are the Oracle database user name and password to use for connecting to the database?

Answer: _____

Is the Oracle Database Client or Instant Client installed on the system where the Secure Agent will run?

Answer: _____

Does the database run in an Oracle Real Application Cluster (RAC)? What's the maximum number of RAC members, including inactive nodes?

Answer: _____

Do you need to capture change data from an Oracle Data Guard logical or physical standby database?

Answer: _____

Do you need to capture change data from tables in a pluggable database (PDB) in an Oracle multitenant environment?

Answer: _____

Do you need to capture change data from tablespaces that use Oracle Transparent Data Encryption (TDE)? If yes, what are the TDE wallet directory and password?

Answer: _____

What is the typical size of units of work (UOWs) for the source tables?

Answer: _____

Oracle redo logs

Are the redo logs in an Oracle Automatic Storage Management (ASM) environment? If you plan to connect to an ASM instance to read redo logs, are you allowed to create a login user ID for ASM that has SYSDBA or SYSASM authority?

Answer: _____

Are ARCHIVELOG mode and minimal global supplemental logging enabled for the Oracle source database? If not, can they be enabled?

Answer: _____

What are the primary and secondary archive log destinations for the archived redo logs from which you want to read change data?

Answer: _____

What is the average amount of archived redo log that is created per hour during peak and non-peak periods for the Oracle database?

Answer: _____

Do you have read access to the redo logs in your environment?

Answer: _____

If you do not have the authority to read the redo logs directly, can the archived redo log files be copied to shared disk or to a file system from which you can access them?

Answer: _____

Do you want Database Ingestion and Replication to read change data from the online log as well as the archived logs?

Answer: _____

Can you make your archived redo logs available for diagnostic use by Informatica Global Customer Support, if necessary, to diagnose an error or anomaly during CDC processing?

Answer: _____

More details for configuring database ingestion

What is the schema name for the source tables from which to replicate data?

Answer: _____

Do you want to replicate data from all tables in the schema or a subset of those tables? If a subset, create a list of them.

Answer: _____

Do the source tables have primary keys? Can supplemental logging be enabled for all of the primary keys?

Answer: _____

Do you have any unkeyed source tables?

Answer: _____

Do the source tables contain columns that have unsupported data types? To determine which data types are not supported for your source types, see the source-specific topics under "Database Ingestion and Replication source considerations" in the Database Ingestion and Replication help.

Answer: _____

Is the default code page of UTF-8 acceptable? If not, which code page do you want to use?

Answer: _____

Do you want to use SSL to encrypt data exchanged between the Secure Agent and database server? Which encryption SSL or TLS protocol do you use?

Answer: _____

Are you allowed to create a new Oracle user and assign the privileges that Database Ingestion and Replication requires to that user? Determine the user name to use.

Answer: _____

Do the source tables contain any Oracle data types that Database Ingestion and Replication does not support?

Answer: _____

Do you want to capture schema drift changes on the source, including add, drop, modify, and rename column operations?

Answer: _____

Oracle privileges

To deploy and run a database ingestion and replication task that has an Oracle source, the source connection must specify a Database Ingestion and Replication user who has the privileges required for the ingestion load type.

Privileges for incremental load processing with log-based CDC

Note: If the Oracle logs are managed by ASM, the user must have SYSASM or SYSDBA authority.

For a database ingestion and replication task that performs an incremental load or combined initial and incremental load using the log-based CDC method, ensure that the Database Ingestion and Replication user (*cmid_user*) has been granted the following privileges:

```
GRANT CREATE SESSION TO <cmid_user>;

GRANT SELECT ON table TO <cmid_user>;     -- For each source table created by user

-- The following grants are required for combined initial and incremental loads only. Do not
-- use ANY TABLE unless your security policy allows it.
GRANT EXECUTE ON DBMS_FLASHBACK TO <cmid_user>;
GRANT FLASHBACK ON table|ANY TABLE TO <cmid_user>;

-- Include the following grant only if you want to Execute the CDC script for enabling
-- supplemental logging from the user interface. If you manually enable supplemental
-- logging, this grant is not needed.
GRANT ALTER table|ANY TABLE TO <cmid_user>;

GRANT SELECT ON DBA_CONSTRAINTS TO <cmid_user>;
GRANT SELECT ON DBA_CONS_COLUMNS TO <cmid_user>;
GRANT SELECT ON DBA_INDEXES TO <cmid_user>;
GRANT SELECT ON DBA_LOG_GROUPS TO <cmid_user>;
GRANT SELECT ON DBA_LOG_GROUP_COLUMNS TO <cmid_user>;
GRANT SELECT ON DBA_OBJECTS TO <cmid_user>;
GRANT SELECT ON DBA_OBJECT_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_TABLESPACES TO <cmid_user>;
GRANT SELECT ON DBA_USERS TO <cmid_user>;

GRANT SELECT ON "PUBLIC".V$ARCHIVED_LOG TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$CONTAINERS TO <cmid_user>;  -- For Oracle multitenant environments
GRANT SELECT ON "PUBLIC".V$DATABASE TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$DATABASE_INCARNATION TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$ENCRYPTION_WALLET TO <cmid_user>;     -- For Oracle TDE access
GRANT SELECT ON "PUBLIC".V$LOG TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$LOGFILE   TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$PARAMETER TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$PDBS TO <cmid_user>;     -- For Oracle multitenant environments
GRANT SELECT ON "PUBLIC".V$SPPARAMETER TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$STANDBY_LOG TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$THREAD TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$TRANSACTION TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$TRANSPORTABLE_PLATFORM TO <cmid_user>;
GRANT SELECT ON "PUBLIC".V$VERSION TO <cmid_user>;

GRANT SELECT ON SYS.ATTRCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CDEF$ TO <cmid_user>;
GRANT SELECT ON SYS.COL$ TO <cmid_user>;
GRANT SELECT ON SYS.COLTYPE$ TO <cmid_user>;
GRANT SELECT ON SYS.IDNSEQ$ TO <cmid_user>;
GRANT SELECT ON SYS.IND$ TO <cmid_user>;
GRANT SELECT ON SYS.INDPART$  TO <cmid_user>;
GRANT SELECT ON SYS.OBJ$ TO <cmid_user>;
GRANT SELECT ON SYS.PARTOBJ$ TO <cmid_user>;
GRANT SELECT ON SYS.RECYCLEBIN$ TO <cmid_user>;
GRANT SELECT ON SYS.TAB$ TO <cmid_user>;
GRANT SELECT ON SYS.TABCOMPART$  TO <cmid_user>;
GRANT SELECT ON SYS.TABPART$ TO <cmid_user>;
GRANT SELECT ON SYS.TABSUBPART$ TO <cmid_user>;
```

```
-- Also ensure that you have access to the following ALL_* views:
ALL_CONSTRAINTS
ALL_CONS_COLUMNS
ALL_ENCRYPTED_COLUMNS
ALL_INDEXES
ALL_IND_COLUMNS
ALL_OBJECTS
ALL_TABLES
ALL_TAB_COLS
ALL_TAB_PARTITIONS
ALL_USERS
```

Privileges for incremental load processing with query-based CDC

For a database ingestion and replication task that performs an incremental load or combined initial and incremental load using the query-based CDC method, ensure that the user has the following privileges at minimum:

```
GRANT CREATE SESSION TO <cmid_user>;

GRANT SELECT ON DBA_INDEXES TO <cmid_user>;
GRANT SELECT ON DBA_OBJECT_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_OBJECTS TO cmid_user;
GRANT SELECT ON DBA_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_USERS TO <cmid_user>;
GRANT SELECT ON DBA_VIEWS TO <cmid_user>;  -- Only if you unload data from views

GRANT SELECT ANY TABLE TO <cmid_user>;
-or-
GRANT SELECT ON table TO <cmid_user>;         -- For each source table created by user

GRANT SELECT ON ALL_CONSTRAINTS TO <cmid_user>;
GRANT SELECT ON ALL_CONS_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_ENCRYPTED_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_IND_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_INDEXES TO <cmid_user>;
GRANT SELECT ON ALL_OBJECTS TO <cmid_user>;
GRANT SELECT ON ALL_TAB_COLS TO <cmid_user>;
GRANT SELECT ON ALL_USERS TO <cmid_user>;

GRANT SELECT ON "PUBLIC"."V$DATABASE"  TO cmid_user;
GRANT SELECT ON "PUBLIC"."V$CONTAINERS" TO cmid_user;
GRANT SELECT ON SYS.ATTRCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CDEF$ TO <cmid_user>;
GRANT SELECT ON SYS.COL$ TO <cmid_user>;
GRANT SELECT ON SYS.COLTYPE$ TO <cmid_user>;
GRANT SELECT ON SYS.IND$ TO <cmid_user>;
GRANT SELECT ON SYS.IDNSEQ$ TO cmid_user;
GRANT SELECT ON SYS.OBJ$ TO <cmid_user>;
GRANT SELECT ON SYS.RECYCLEBIN$ TO <cmid_user>;
GRANT SELECT ON SYS.TAB$ TO <cmid_user>;
```

Privileges for initial load processing

For a database ingestion and replication task that performs an initial load, ensure that the user has the following privileges at minimum:

```
GRANT CREATE SESSION TO <cmid_user>;

GRANT SELECT ON DBA_INDEXES TO <cmid_user>;
GRANT SELECT ON DBA_OBJECT_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_OBJECTS TO cmid_user;
GRANT SELECT ON DBA_TABLES TO <cmid_user>;
GRANT SELECT ON DBA_USERS TO <cmid_user>;
GRANT SELECT ON DBA_VIEWS TO <cmid_user>;  -- Only if you unload data from views

GRANT SELECT ANY TABLE TO <cmid_user>;
-or-
GRANT SELECT ON table TO <cmid_user>;         -- For each source table created by user
```

```
GRANT SELECT ON ALL_CONSTRAINTS TO <cmid_user>;
GRANT SELECT ON ALL_CONS_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_ENCRYPTED_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_IND_COLUMNS TO <cmid_user>;
GRANT SELECT ON ALL_INDEXES TO <cmid_user>;
GRANT SELECT ON ALL_OBJECTS TO <cmid_user>;
GRANT SELECT ON ALL_TAB_COLS TO <cmid_user>;
GRANT SELECT ON ALL_USERS TO <cmid_user>;

GRANT SELECT ON "PUBLIC"."V$DATABASE"  TO cmid_user;
GRANT SELECT ON "PUBLIC"."V$CONTAINERS" TO cmid_user;
GRANT SELECT ON SYS.ATTRCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CDEF$ TO <cmid_user>;
GRANT SELECT ON SYS.COL$ TO <cmid_user>;
GRANT SELECT ON SYS.COLTYPE$ TO <cmid_user>;
GRANT SELECT ON SYS.IND$ TO <cmid_user>;
GRANT SELECT ON SYS.IDNSEQ$ TO cmid_user;
GRANT SELECT ON SYS.OBJ$ TO <cmid_user>;
GRANT SELECT ON SYS.RECYCLEBIN$ TO <cmid_user>;
GRANT SELECT ON SYS.TAB$ TO <cmid_user>;n
```

Oracle privileges for Amazon RDS for Oracle sources

If you have an Amazon RDS for Oracle source, you must grant certain privileges to the Database Ingestion and Replication user.

Important: You must log in to Amazon RDS under the master username to run GRANT statements and procedures.

Grant the SELECT privilege, at minimum, on objects and system tables that are required for CDC processing to the Database Ingestion and Replication user (*cmid_user*). Some additional grants are required in certain situations.

Use the following GRANT statements:

```
GRANT SELECT ON "PUBLIC"."V$ARCHIVED_LOG"           TO "cmid_user";

GRANT SELECT ON "PUBLIC"."V$DATABASE"               TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$LOG"                    TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$LOGFILE"                TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$TRANSPORTABLE_PLATFORM" TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$THREAD"                 TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$DATABASE_INCARNATION"   TO "cmid_user";
GRANT SELECT ON "PUBLIC"."V$TRANSACTION"            TO "cmid_user";

GRANT SELECT ON "SYS"."DBA_CONS_COLUMNS"      TO "cmid_user";
GRANT SELECT ON "SYS"."DBA_CONSTRAINTS"       TO "cmid_user";
GRANT SELECT ON DBA_INDEXES                   TO "cmid_user";
GRANT SELECT ON "SYS"."DBA_LOG_GROUP_COLUMNS" TO "cmid_user";
GRANT SELECT ON "SYS"."DBA_TABLESPACES"       TO "cmid_user";

GRANT SELECT ON "SYS"."OBJ$"          TO "cmid_user";
GRANT SELECT ON "SYS"."TAB$"          TO "cmid_user";
GRANT SELECT ON "SYS"."IND$"          TO "cmid_user";
GRANT SELECT ON "SYS"."COL$"          TO "cmid_user";

GRANT SELECT ON "SYS"."PARTOBJ$"      TO "cmid_user";
GRANT SELECT ON "SYS"."TABPART$"      TO "cmid_user";
GRANT SELECT ON "SYS"."TABCOMPART$"   TO "cmid_user";
GRANT SELECT ON "SYS"."TABSUBPART$"   TO "cmid_user";
COMMIT;

/* For combined load jobs:*/
GRANT EXECUTE ON DBMS_FLASHBACK TO "cmid_user";

/*To provide read access to the Amazon RDS online and archived redo logs:*/
```

```
GRANT READ ON DIRECTORY ONLINELOG_DIR TO "cmid_user";
GRANT READ ON DIRECTORY ARCHIVELOG_DIR TO "cmid_user";
```

Additionally, log in as the master user and run the following Amazon RDS procedures to grant the SELECT privilege on some more objects:

```
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_TABLES',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_OBJECTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_OBJECT_TABLES',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_VIEWS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
/begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_USERS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$CONTAINERS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$PARAMETER',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$SPPARAMETER',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$STANDBY_LOG',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
```

```
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$VERSION',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_CONS_COLUMNS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_CONSTRAINTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_OBJECTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_TABLES',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_TAB_PARTITIONS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_USERS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rd 'ALL_TABLES',
p_grantee => 'sadmin_util.grant_sys_object(
p_obj_name => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_TAB_PARTITIONS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ATTRCOL$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
```

```
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'CCOL$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'COLTYPE$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'INDPART$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'IDNSEQ$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'CDEF$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'RECYCLEBIN$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
/* Only required for RDS21 which supports PDB*/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$PDBS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
```

Oracle log access methods for CDC

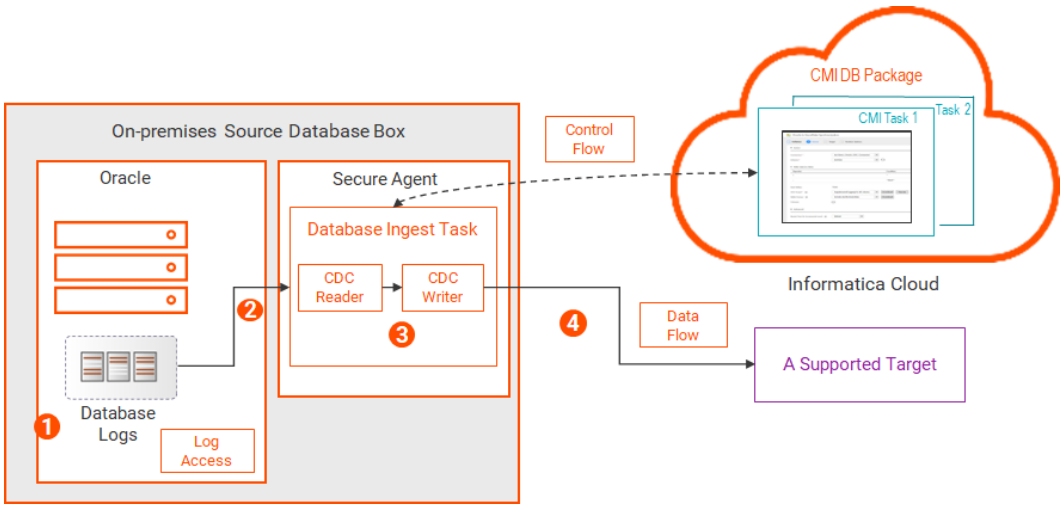
Database ingestion and replication incremental load and combined initial and incremental load jobs can access the Oracle redo logs for CDC processing in alternative ways, depending on your environment and requirements.

Direct log access

Database ingestion and replication jobs can directly access the physical Oracle redo logs on the on-premises source system to read change data.

Note: If you store the logs on solid-state disk (SSD), this method can provide the best performance.

The following image shows the data flow:

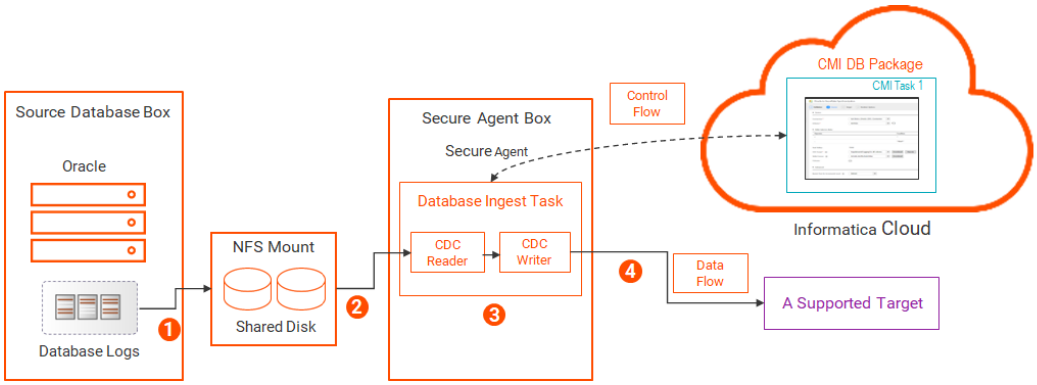


1. The Oracle database writes change records to the database log files on disk.
2. The Database Ingestion and Replication CDC Reader reads the physical log files and extracts change records from the log files for the source tables of CDC interest.
3. The Database Ingestion and Replication CDC Writer reads the change records.
4. The CDC Writer applies the change records to the target.

NFS-mounted logs

Database ingestion and replication jobs can access to Oracle database logs from shared disk by using a Network File Sharing (NFS) mount or another method such as Network Attached Storage (NAS) or clustered storage.

The following image shows the data flow:



1. The Oracle database writes change records to database log files. The log files are written to shared disk. The shared disk can be on any system that allows the files to appear as local to both the database and Secure Agent hosts. This sharing can be achieved by using NFS, as shown above, or by using Network Attached Storage (NAS) or clustered storage.
2. The Database Ingestion and Replication CDC Reader reads the log files from the NFS server over the network and extracts the change records for the source tables of CDC interest.
3. The Database Ingestion and Replication CDC Writer reads the change records.
4. The CDC Writer applies the change records to the target.

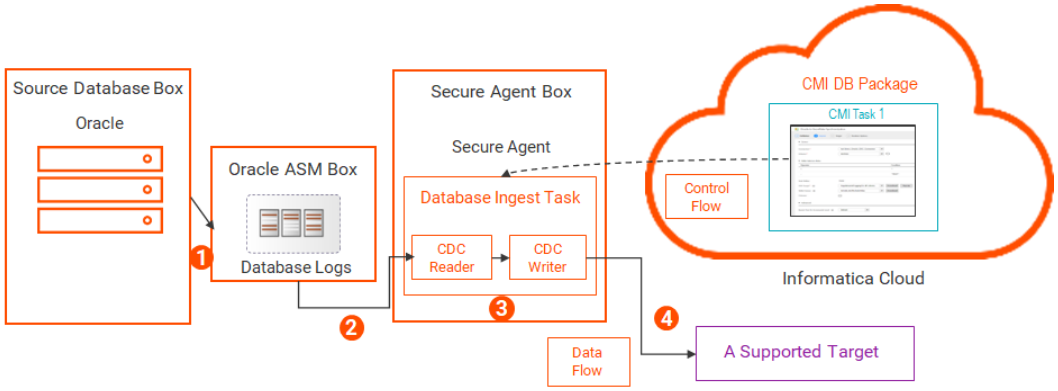
ASM-managed logs

Database ingestion and replication jobs can access Oracle redo logs that are stored in an Oracle Automatic Storage Management (ASM) system. To read change data from the ASM-managed redo logs, the ASM user must have SYSASM or SYSDBA authority on the ASM instance.

When you configure an Oracle Database Ingestion connection, complete the properties that include "ASM" in their names.

Also, Informatica recommends that you set the sqlnet.recv_timeout parameter in the local sqlnet.ora file to less than 5 minutes when reading data from redo log files in Oracle ASM. This parameter specifies the duration of time that the Oracle client waits for a response from ASM before a query times out. Network interrupts and other factors can occasionally make Oracle connections unresponsive. Setting this value ensures that the reader can respond and recover from any such situation in a timely manner.

The following image shows the data flow:

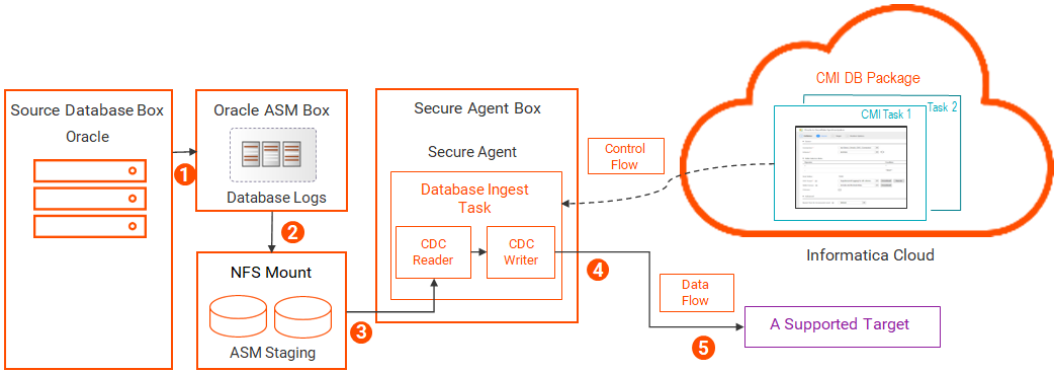


1. The Oracle database writes change records to the ASM-managed database log files.
2. The Database Ingestion and Replication CDC Reader reads the ASM-managed log files and extracts the change records for the source tables of CDC interest.
3. The Database Ingestion and Replication CDC Writer reads the change records.
4. The CDC Writer applies the change records to the target.

ASM-managed logs with a staging directory

Database ingestion jobs can access ASM-managed redo logs from a staging directory in the ASM environment. In comparison to using ASM only, this method can provide faster access to the log files and reduce I/O on the ASM system. To read change data from the ASM-managed logs, the ASM user must have SYSASM or SYSDBA authority on the ASM instance.

The following image shows the data flow:

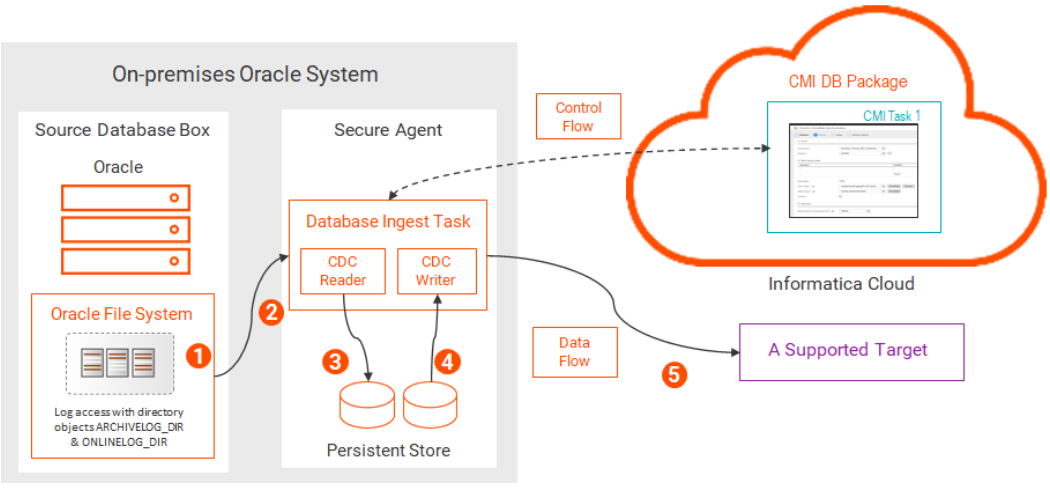


1. The Oracle database writes change records to the ASM-managed log files.
2. ASM copies the logs to a staging directory.
   The staging directory must be on shared disk, such as an NFS mount, so that ASM can write data to it and database ingestion and replication jobs can read data from it.
3. The Database Ingestion and Replication CDC Reader reads the log files in the staging directory and extracts the change records for the source tables of CDC interest.
4. The Database Ingestion and Replication CDC Writer reads the change records.
5. The CDC Writer applies the change records to the target.

BFILE access to logs in the Oracle server file system by using directory objects

On an on-premises Oracle source system, you can configure Database Ingestion and Replication to read online and archived redo logs from the local Oracle server file system by using Oracle directory objects with BFILE locators. You must create Oracle directory objects named ARCHIVELOG_DIR and ONLINELOG_DIR that point to the locations of the Oracle redo log files. For information about configuring BFILE access, see "Configuring BFILE access to Oracle redo logs in the Oracle file system" on page 57.

The following image shows the data flow:



1. The Oracle database writes change records to the redo log files in the local Oracle server file system. When a database ingestion and replication task needs to read log files, it connects to Oracle and issues a select request that references the ARCHIVELOG_DIR or ONLINELOG_DIR directory object to access the logs.
2. The Database Ingestion and Replication CDC Reader reads the log file and extracts the change records for the source tables of CDC interest.

3. The Database Ingestion and Replication CDC Writer reads the change records.
4. The CDC Writer applies the change records to the target.

Configuring BFILE access to Oracle redo logs in the Oracle file system

If you store redo logs in the local Oracle server file system and want to access the logs by using Oracle directory objects with BFILEs, perform the following configuration tasks:

Complete the following usual Oracle source preparation tasks, which are not specific to BFILE access:

- Define the ORACLE_HOME environment variable on the Linux or Windows system where the Secure Agent runs for Database Ingestion and Replication to use the Oracle Call Interface (OCI) to communicate with the Oracle source database.
- Make sure the Database Ingestion and Replication user has the Oracle privileges that are required for the database ingestion and replication incremental load processing. For more information, see "Oracle privileges" on page 48.
- Enable ARCHIVELOG mode for the Oracle database.
- Define the archive log destination.

  Note: For BFILE access, use a specific archive log destination directory. Do not use the Oracle USE_DB_RECOVERY_FILE_DEST keyword to store archive logs by date in a Fast Recovery Area (FRA).
- Enable Oracle minimal global supplemental logging on the source database.
- If your Oracle source tables have primary keys, ensure that supplemental logging is enabled for all primary key columns. For source tables that do not have primary keys, ensure that supplemental logging is enabled for all columns from which change data will be captured.

  Note: When you create a database ingestion and replication task, you have the option of generating a script that implements supplemental logging for all columns or only primary key columns for the selected source tables.
- Ensure that the Oracle MAX_STRING_SIZE initialization parameter is *not* set to EXTENDED. If it is set to EXTENDED, Database Ingestion and Replication will not be able to replicate inserts and updates for tables containing columns defined with large (extended size) VARCHAR2, NVARCHAR2, or RAW columns.

Additionally, for BFILE access, perform the following steps:

1. Query the Oracle database for the online and archived redo log locations in the Oracle server file system. You can use the following example queries:
   To get location of the online redo logs:

   ```
   select * from v$logfile;
   ```

   To get the archive log destination:

   ```
   select dest_id, dest_name, destination, status from V$ARCHIVE_DEST;
   ```
2. Create the ONLINELOG_DIR and ARCHIVELOG_DIR directory objects that point to the locations of log files from step 1. An Oracle directory object specifies a logical alias name for a physical directory on the Oracle server file system under which the log files to be accessed are located. For example:

   ```
   CREATE DIRECTORY ONLINELOG_DIR AS '/u01/oracle/data';
   CREATE DIRECTORY ARCHIVELOG_DIR AS '/u01/oracle/archivedata';
   ```

   Note: If you plan to set the reader mode to ARCHIVEONLY in the Oracle Database Ingestion connection to read changes only from archive logs, you do not need to create an ONLINELOG_DIR directory or directory object.

   The Oracle database does not verify that the directories you specify exist. Make sure that you specify valid directories that exist in the Oracle file system.

3. To verify that the directory objects were created with the correct file system paths for the redo logs, issue a select statement such as:

```
select * from all_directories;
OWNER    DIRECTORY_NAME      DIRECTORY_PATH
-------- ------------------- --------------------------------
SYS      ARCHIVELOG_DIR      /u01/oracle/data/JO112DTL
SYS      ONLINELOG_DIR       /u01/oracle/data/JO112DTL
```

4. Grant read access on the ONLINELOG_DIR and ARCHIVELOG_DIR directory objects to the Database Ingestion and Replication user who is specified in the Oracle Database Ingestion connection properties. For example:

```
grant read on directory "ARCHIVELOG_DIR" to "cmid_user";
grant read on directory "ONLINELOG_DIR" to "cmid_user";
```

   Note: If the ONLINELOG_DIR path does not exist or match the path to the active redo logs, Database Ingestion and Replication tries to create the directory. If you do not have sufficient privileges to create the directory object, an error message is issued. In this case, ask your DBA to create the directory object with the correct path.

5. In the Oracle Database Ingestion connection properties, select the **BFILE Access** check box.

Oracle Data Guard databases or far sync instances as sources

Database Ingestion and Replication can capture change data from Oracle Data Guard primary databases, logical or physical standby databases, and far sync instances.

A far sync instance is a remote Oracle Data Guard destination that accepts redo from the primary database and then ships that redo to other members of the Oracle Data Guard configuration.

You can initially load a target with data either from the Oracle Data Guard primary database or from a standby database that is open in read mode.

Configuration

Oracle change capture configuration depends on the Oracle Data Guard database type.

- For the *primary Oracle database* , grant SELECT permissions on the V$STANDBY_LOG view to the Database Ingestion and Replication user:

  ```
  GRANT SELECT ON "PUBLIC".V$STANDBY_LOG TO <cmid_user>;
  ```

  If the primary database is in an Amazon RDS for Oracle environment:

  ```
  begin
  rdsadmin.rdsadmin_util.grant_sys_object(
  p_obj_name => 'V_$STANDBY_LOG',
  p_grantee => 'cmid_user',
  p_privilege => 'SELECT');
  end;
  ```

- For a *physical standby database in mount mode* (not open with read only access), set the following Oracle Database Ingestion connection properties:
  - Database Connect String - Ensure that it points to the primary database to read the Oracle catalog.
  - Standby Connect String - An Oracle connection string, defined in TNS, that the log reader uses to connect to the Oracle physical standby database and monitor the logs.
  - Standby User Name - A user ID that the log reader uses to connect to the Oracle physical standby database. This user ID must have SYSDBA authority.
  - Standby Password - A password that the log reader uses to connect to the Oracle physical standby database.

  Note: With a database in mount mode, you can use a password file for user authentication. Initially, you must grant SYSDBA authority to the user. If you want to avoid granting permanent SYSDBA authority to

the user, you can copy the primary password file to the physical standby or far sync instance and then revoke SYSDBA authority for the user. Repeat this process whenever you refresh the password file.

Optionally, configure the following additional connection properties:

- RAC Members - The maximum number of active threads on the Data Guard primary database when the database is in a RAC environment,
- Reader Standby Log Mask - A mask that the log reader uses for selecting redo logs for an Oracle standby database when the database uses multiplexing of redo logs.

For more information, see "Oracle Database Ingestion connection properties" in *Connectors and Connections*.

- For a *logical standby database* , no special configuration tasks are required. Configure it the same way as for an Oracle database that is not in a Data Guard environment.

Standby-to-primary role transitions

In an Oracle Data Guard environment, a physical standby database can transition to the primary role. Usually, the role transition occurs because of a failover or switchover. During the transition, all active connections to the physical standby database terminate.

To be able to resume CDC processing after the physical standby database transitions to the primary role, you might need to adjust some connection configuration properties on the original standby system for Database Ingestion and Replication to process past the transition point. After the transition, you can adjust the properties again for optimal performance in the new primary database environment.

The following table describes these connection properties by transition phase:

Connection Property	Before Transition	During Transition	After Transition
RAC Members	Specify the number of active threads on the primary database.	Specify the total number of active threads with unique thread IDs on *both* the standby database and primary database. For example, if the primary database is a two-node RAC database that uses thread IDs 1 and 2 and the standby database is a 3-node RAC database that uses thread IDs 2, 3, and 4, specify a property value of 4.	After the restart point has progressed beyond the transition point, edit the property value, as needed, for optimal performance of change data capture from the new primary database. Informatica recommends that you use the lowest value that is suitable for your environment to minimize the overhead of CDC thread tracking.
Reader Standby Log Mask Standby Connect String Standby User Name Standby Password	Remove all standby properties. They're not applicable to physical standby databases open for read only access.	Properties remain removed.	Do not specify these properties. They're not used for a primary database.
Database Connect String	If the standby database is not open, define the connection string for the primary database. If the standby database is open, define the connection string for the standby database.	Specify the connection string for the database that will have the primary role after the role transition.	Ensure that this connection property defines the connection string for the new primary database.

Oracle archive log retention considerations

Database ingestion and replication incremental load and combined initial and incremental load jobs must be able to access transaction data in Oracle online and archive redo logs. If the logs are not available, database ingestion and replication jobs end with an error.

Typically, the Oracle DBA sets the archive log retention period based on your oganization's particular business needs and Oracle environment. Make sure that the source archive logs are retained for the longest period for which you expect change capture to be stopped or latent, plus about 1 hour, so that the logs will be available for restart processing.

To determine if the current log retention policy in your environment is sufficient to accommodate database ingestion and replication change capture processing, consider the following factors:

- How long are Oracle transactions typically open on a source?
- What is the longest period of time that change capture is allowed to be down or latent, accounting for weekends and holidays?
- What is the replication latency from source to target?
- Do you run database ingestion and replication jobs based on a schedule? If yes, what type of schedule?

- Is the pwx.cdcreader.oracle.option.additional ageOutPeriod=*minutes* custom property set on the **Source** page of task wizard?

  Note: This property specifies the age at which outstanding UOWs without change records of CDC interest are removed from the calculation of the next restart point. You can use the property to prevent CDC failures that might occur if you shut down and then restart capture processing while the transaction is outstanding and the redo log in which the UOW started is not available.
- What is the redo generation rate?
- Do you ship copies of archive logs to a secondary system?

If the archive logs are not available when you need to restart capture processing in the logs, you can ask your DBA to restore them and to modify the retention period if necessary. Otherwise, perform another initial load to re-materialize the target and then start incremental change data processing again. However, in this case, you might lose some changes.

PostgreSQL sources

To use PostgreSQL sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations.

Source preparation

To use PostgreSQL sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations.

On the Secure Agent system, install the ODBC driver that is appropriate for your operating system.

- On Windows, install the latest version of the 64-bit PostgreSQL ODBC driver.
  1. Download and install the PostgreSQL ODBC driver.

     Note: If the source database contains objects with multibyte-character names, such as table names, column names, and publication names, you must use either a PostgreSQL Unicode ODBC driver or the DataDirect ODBC for PostgreSQL driver. This requirement applies to all PostgreSQL source types, including Amazon Aurora PostgreSQL, Azure Database for PostgreSQL - Flexible Server, Cloud SQL for PostgreSQL, and RDS for PostgreSQL. If you do not use a Unicode-compatible ODBC driver, your incremental load jobs will fail when encountering a multibyte-character name.
  2. Set the PGSQL_ODBC_DRIVER environment variable to the driver name that is displayed by ODBC Data Source Administrator (64-bit).

     Note: You can override this driver for a database ingestion and replication task by setting the pwx.custom.pgsql_odbc_driver custom property on the Source page of the task wizard.
- On Linux or UNIX, the DataDirect ODBC driver for PostgreSQL is delivered as part of the Linux installation. Alternatively, you can install the unixODBC or iODBC driver manager or the PostgreSQL ODBC driver.
  1. If you do not want to use the DataDirect ODBC for PostgreSQL driver that is provided in the Linux installation, install the unixODBC or iODBC driver manager or install the PostgreSQL ODBC driver.

     Note: If the source database contains objects with multibyte-character names, such as table names, column names, and publication names, you must use either a PostgreSQL Unicode ODBC driver or the DataDirect ODBC for PostgreSQL driver. This requirement applies to all PostgreSQL source types, including Amazon Aurora PostgreSQL, Azure Database for PostgreSQL - Flexible Server, Cloud SQL for PostgreSQL, and RDS for PostgreSQL. If you do not use a Unicode-compatible ODBC driver, your incremental load jobs will fail when encountering a multibyte-character name.

2. Add a PostgreSQL entry to odbcinst.ini.

```
[PGSQL]
Description = ODBC for PostgreSQL
Driver =
Setup =
Driver64 = /usr/pgsql-9.6/lib/psqlodbca.so
Setup64 = /usr/lib64/libodbcpsqlS.so
FileUsage = 1
```

3. Optional. Set the following environment variables:

   - Set the ODBCSYSINI variable to the directory where odbcinst.ini is located. If odbcinst.ini is located in the default /etc directory, you do not need to set the ODBCSYSINI variable.
   - Add the directory where the PostgreSQL ODBC driver is installed to the LD_LIBRARY_PATH variable. If the driver is installed in the default directory of /usr/lib64, you do not need to add the path to the LD_LIBRARY_PATH variable.
   - Set the PGSQL_ODBC_DRIVER parameter to the driver name that you specified in odbcinst.ini.

   For example:

```
export ODBCSYSINI=/root/infaagent
export LD_LIBRARY_PATH=/usr/pgsql-9.6/lib
export PGSQL_ODBC_DRIVER=PGSQL
```

On the *PostgreSQL database system*, perform the following configuration steps:

1. For incremental load and initial and incremental load jobs, ensure that the PostgreSQL postgresql.conf configuration file specifies the wal_level=logical parameter.
   This parameter determines how much information PostgreSQL writes to the Write-Ahead Log (WAL). The setting of logical adds information that is required to support logical decoding.

   To set wal_level to logical on Amazon Aurora PostgreSQL or Amazon RDS for PostgreSQL sources, set the rds.logical_replication parameter to 1 in the cluster parameter group. For Azure Database for PostgreSQL - Flexible Server, set the wal_level parameter to logical on the Server Parameters page in the Azure portal.

   For Cloud SQL for PostgreSQL sources, perform the following actions:

   a. Connect to the database by using the public IP.

      Note: Ensure that you add the required IPs under **Authorized networks** in the Google Cloud console.

   b. Create a Cloud SQL for PostgreSQL database instance replica.

   c. In the Cloud Shell, as a local admin user, run the following commands:

```
alter database postgres set default_transaction_read_only = off;
gcloud sql connect database_replica --user=postgres --quiet;
ALTER USER postgres WITH REPLICATION;
CREATE USER replication_user WITH REPLICATION IN ROLE cloudsqlsuperuser LOGIN
PASSWORD 'password';
ALTER USER postgres WITH REPLICATION;
```

   d. In the Google Cloud console, add the following database flags:

      - **cloudsql.logical_decoding**. Set the value to `on`.
      - **max_replication_slots**. Set the value to `64`.
      - **cloudsql.enable_pglogical**. Set the value to `on`.
      - **max_wal_senders**. Set the value to `64`.

   e. Restart the database instance.

2. If you use the DataDirect ODBC for PostgreSQL driver, ensure that the database does not use the SCRAM-SHA-256 authentication method. Use another authentication method, such as MD5.

Note: The PostgreSQL ODBC driver supports the SCRAM-SHA-256 authentication method. In PostgreSQL 13, this authentication method became the default method.

3. To deploy and run a database ingestion and replication task that includes a PostgreSQL source, the source connection must specify a database user who has the required privileges. Create the user and grant privileges to that user in the following ways:
   - For initial load jobs, use the following SQL statements:

     ```
     CREATE USER dbmi_user WITH PASSWORD 'password';
     GRANT SELECT ON ALL TABLES IN SCHEMA schema TO dbmi_user;
     ```

   - For incremental load and initial and incremental load jobs with on-premises PostgreSQL sources, use the following SQL statement:

     ```
     CREATE USER dbmi_user WITH PASSWORD 'password' REPLICATION;
     ```

     For Amazon Aurora PostgreSQL and RDS for PostgreSQL sources, use the following statements:

     ```
     CREATE USER dbmi_user WITH PASSWORD 'password';
     GRANT rds_replication to dbmi_user;
     ```

     Additionally, if you use the pgoutput plugin, use the following SQL statement to grant ownership of the tables in the database that you want to add to the pgoutput publication to the *dbmi_user* that you created:

     ```
     GRANT CREATE ON DATABASE database TO dbmi_user;
     ```

4. If you plan to use the wal2json plugin for logical decoding output for incremental load or initial and incremental load jobs, install the plugin.
5. If you plan to use the pgoutput plugin for incremental load or initial and incremental load jobs, use the following SQL statement to create publications for database ingestion jobs:

   ```
   CREATE PUBLICATION publication_name [FOR TABLE [ONLY] table_name [*] [,...] | FOR
   ALL TABLES ];
   ```

   Ensure that the publication includes all tables that you want to replicate to the target.
6. For incremental load and initial and incremental load jobs, use the following function to create a PostgreSQL logical replication slot:

   ```
   SELECT pg_create_logical_replication_slot('slot_name', 'plugin_type');
   ```

   Where the *plugin_type* is either the pgoutput plugin or the wal2json plugin.
7. For incremental load and initial and incremental load jobs with PostgreSQL 9.6 sources, ensure that the max_replication_slots parameter in the postgresql.conf configuration file has a value greater than or equal to the number of concurrent database ingestion jobs that you plan to use.

   Important: All replication slots must be unique across all concurrent jobs.
8. For incremental load and initial and incremental load jobs, ensure that the PostgreSQL sources use the UTF-8 encoding.
9. Ensure that the PostgreSQL sources use the UTF-8 encoding. If you use another encoding for the source database, your initial load, incremental load, and combined initial and incremental load jobs might fail.

Usage considerations

- Database Ingestion and Replication supports the following types of PostgreSQL sources in database ingestion and replication jobs of any load type: on-premises PostgreSQL, Amazon Aurora PostgreSQL, Azure Database for PostgreSQL - Flexible Server, Cloud SQL for PostgreSQL, and RDS PostgreSQL.
- Database ingestion and replication jobs that have a PostgreSQL source can have any target type except for PostgreSQL.
- Database Ingestion and Replication supports Cloud SQL for PostgreSQL sources of any load type that have a Google BigQuery or Snowflake target only.

- Because Database Ingestion and Replication expects each source table row to be unique, Informatica recommends that each source table have a primary key. Database Ingestion and Replication does not honor unique indexes in place of a primary key. If no primary key is specified, Database Ingestion and Replication treats all columns as if they are part of the primary key.
- Database Ingestion and Replication supports schema drift options for PostgreSQL sources in database ingestion and replication incremental load and initial and incremental load jobs with the following limitations:
  - PostgreSQL does not support changes to primary keys for tables from which change data capture is enabled.
  - Database ingestion and replication jobs cannot capture DML changes from source tables for which table partition IDs are changed.
- Database Ingestion and Replication does not support generated columns in incremental load jobs that have a PostgreSQL source. If the source table contains generated columns, change data capture will ignore them and will continue with the rest of the columns.
- Database ingestion and replication jobs can replicate data from PostgreSQL BYTEA, JSON, JSONB, TEXT, and XML columns if you select **Include LOBs** under **Advanced** on the **Source** page of the task wizard. The supported target types depend on the load type:
  - For initial load jobs and incremental load jobs: Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, Microsoft Azure Synapse Analytics, Oracle, Oracle Cloud Object Storage, Snowflake, and SQL Server.
  - For incremental load jobs: Azure Event Hubs and SQL Server.
  - For combined initial and incremental load jobs: Snowflake and SQL Server.

  Database ingestion and replication incremental load and initial and incremental load jobs can replicate data from TEXT, XML, and BIT VARYING and CHARACTER VARYING columns without length limits if you set the pwx.custom.pgsql_enable_lobs custom property to true on the **Source** page of the task wizard. Incremental load and initial and incremental load jobs also always replicate data from BYTEA, JSON, and JSONB columns.

  LOB column data is truncated before being written to the target if it is greater in size than a byte limit that depends on the LOB type and target type. For more information, see "Configuring the source" on page 106.
- For PostgreSQL 9.6, the pgoutput plugin is not available.
- For initial load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types:
  - ABSTIME
  - Array types
  - NAME
  - Object identifier types
  - PG_LSN
  - RELTIME
  - Text search types:
    - TSQUERY
    - TSVECTOR
  - User-defined types

For incremental load and initial and incremental load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types, in addition to those not supported for initial load jobs:

- Spatial types
  - Box
  - Circle
  - Line
  - LSeg
  - Path
  - Point
  - Polygon
- Unbounded varying types

Database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have these data types.

For information about the default mappings of supported PostgreSQL data types to target types, see "Default Data Type Mappings" on page 172.

- If you update the primary key value in the PostgreSQL source table for a record that does not exist in the target table, the record is not replicated to the target. However, the monitoring interface increments the Update count to include the primary key update. The data will be replicated to the target only if the record exists in the target table before the primary key update occurs.

SAP HANA and SAP HANA Cloud sources

To use SAP HANA and SAP HANA Cloud sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations.

Source preparation

- The SAP HANA Database Ingestion connector uses JDBC to connect to the SAP HANA and SAP HANA Cloud database to read data and metadata and to test connection properties. You must download the SAP HANA JDBC driver file, ngdbc.jar, and copy it to a specific subdirectory of the Secure Agent installation directory on the machine where the Secure Agent runs.
  1. Download the SAP HANA JDBC driver jar file, ngdbc.jar, to the Linux or Windows machine where the Secure Agent runs.
     Verify that you download the most recent version of the file. If you encounter any issues with downloading the file, contact SAP Customer Support.
  2. Copy the ngdbc.jar file to the following directory:

     ```
     <Secure Agent installation directory>/ext/connectors/thirdparty/informatica.hanami
     ```
  3. Restart the Secure Agent.
- Create a Database Ingestion and Replication user. Connect to the source database as a user with admin authority and execute the following statement:

  ```
  CREATE USER dbmi_user password "<password>" NO FORCE_FIRST_PASSWORD_CHANGE;
  ```

  This statement creates a user in the database with the default permissions, which permit basic data dictionary views to be read and the required CDC objects to be created in the user's own schema.

- To deploy and run a database ingestion and replication task that includes an SAP HANA or SAP HANA Cloud source, the source connection must specify a Database Ingestion and Replication user (*dbmi_user*) who has the privileges to read metadata and other information from the following system views:
  - SYS.M_DATABASE - Used to fetch the database version.
  - SYS.M_CS_PARTITIONS - Used to identify if a table is partitioned. (Not applicable for SAP HANA Cloud)
  - SYS.SCHEMAS - Used to fetch the list of schemas for a database.
  - SYS.TABLES - Used to fetch a list of table names for a schema.
  - SYS.TABLE_COLUMNS - Used to fetch column metadata for a table.
  - SYS.INDEXES - Used to fetch index information for a table.
  - SYS.INDEX_COLUMNS - Used to fetch index information for a table.
- For incremental load jobs, grant the following privileges:
  - For triggers to write rows to the PKLOG and shadow _CDC tables of the *dbmi_user*, grant INSERT access on the *dbmi_user*'s schema to the user that owns the schema of the source tables (*schema_user*):

    ```
    GRANT INSERT ON SCHEMA dbmi_user TO schema_user;
    ```
  - To capture change data from source tables in the schema_user's schema by using triggers, execute one of the following statements:

    ```
    GRANT TRIGGER ON SCHEMA schema_user TO dbmi_user;
    ```

    This statement grants trigger access to all tables in the schema.

    - or -

    ```
    GRANT TRIGGER ON database.table_name TO dbmi_user;
    ```

    This statement grants trigger access to a specific source table. Use this statement when you want to capture data from just a few selected tables. Repeat the grant for each source table of CDC interest.
- For initial load jobs, execute one of the following grant statements to read data from source tables:

  ```
  GRANT SELECT ON SCHEMA schema_user TO dbmi_user;
  ```

  This statement grants SELECT access to all tables in the schema.

  - or -

  ```
  GRANT SELECT ON database.table_name TO dbmi_user;
  ```

  This statement grants SELECT access on a specific source table. Repeat this grant for each source table from which you want to read data.
- For SAP HANA Cloud sources, the connection requires an SAP HANA JDBC custom connection property setting for encryption. Enter the following properties in the **Advanced Connection Properties** field in the SAP HANA Database Ingestion connection properties:

  ```
  encrypt=true&validateCertificate=false
  ```

Usage considerations

- Database Ingestion and Replication supports SAP HANA and SAP HANA Cloud sources on Red Hat Linux or SUSE Linux for initial load and incremental load jobs but not for combined initial and incremental load jobs.
- Database ingestion and replication incremental load jobs support table names up to 120 characters in length.
- Schema drift options are not supported for incremental load jobs with SAP HANA or SAP HANA Cloud sources.
- Database Ingestion and Replication does not require primary keys on the SAP HANA source tables for initial load or incremental load jobs.

- Database Ingestion and Replication does not support the following source data types, even though they're mapped to default column data types on the target:

  - ARRAY
  - BINTEXT
  - BLOB
  - CLOB
  - NCLOB
  - ST_GEOMETRY
  - ST_POINT
  - TEXT

  Database Ingestion and Replication jobs propagate nulls for columns that have these data types.

  Note: The ALPHANUM, BINTEXT, CHAR and CLOB data types are not available in SAP HANA Cloud.

  For information about the default mappings of SAP HANA data types to target data types, see "Default Data Type Mappings" on page 172.

- For incremental load jobs, Database Ingestion and Replication requires the following tables in the source database:

  - PKLOG log table. Contains metadata about captured DML changes, such as the change type and timestamp, transaction ID, schema name, and table name.
  - PROCESSED log table. Contains the maximum sequence number (SCN) for the most recent change data capture cycle.
  - Shadow <*schema*>.<*tablename*>_CDC tables. Contains before images of updates and after images of inserts, updates, and deletes captured from the source tables, with metadata such as the transaction ID and timestamp. A shadow table must exist for each source table from which changes are captured.

  Also, Database Ingestion and Replication uses AFTER DELETE, AFTER INSERT, and AFTER UPDATE triggers to get before images and after images of DML changes for each source table and to write entries for the changes to the PKLOG and shadow _CDC tables. Database Ingestion and Replication also writes SAP HANA sequence values to the PKLOG and shadow _CDC tables for each insert, update, and delete row processed. The sequence values link the rows of the shadow _CDC table to the rows of the PKLOG table during CDC processing.

  When you deploy a task, Database Ingestion and Replication validates the existence of the PKLOG, PROCESSED, and shadow _CDC tables, the triggers, and the sequences. The deploy operation fails if these items do not exist.

- From the **Source** page in the task wizard, you can download or execute a CDC script that creates the PKLOG, PROCESSED, and shadow _CDC tables, the triggers, and the sequences. If you specified a **Trigger Prefix** value in the SAP HANA Database Ingestion connection properties, the names of the generated triggers begin with *prefix*_.

  By default, the triggers capture the application's system user. If you want to capture the transaction user instead, download the CDC script and replace all occurrences of 'APPLICATIONUSER' with 'XS_APPLICATIONUSER' in the script. For example, you could make this substitution in the AFTER DELETE triggers so that you can identify and filter out deletes associated with an archiving process.

- In incremental load jobs, only source table columns that are included in the shadow _CDC tables are part of the source schema definition. When a new column is added to the source table and is not present in the shadow _CDC tables, the newly added column is ignored.

- If you remove or rename an existing source column, modify the triggers and the corresponding shadow _CDC table accordingly. Otherwise, the job will fail.

- During the execution of an incremental load job, some housekeeping takes place to delete outdated records from the PKLOG and shadow _CDC tables to maintain the size of the tables. To enable automatic housekeeping of the PKLOG table and shadow _CDC tables, specify a value greater than 0 in the **Log Clear** field in the SAP HANA Database Ingestion connection properties. The default value is 14 days, and the maximum value is 366. A value of 0 deactivates housekeeping.
  Housekeeping takes place while an incremental load job is running. If multiple jobs are running against different tables, each job performs housekeeping against the PKLOG table and against the shadow _CDC tables that are defined for that job only. If you remove a source table from the job, no purging for the corresponding shadow _CDC table occurs.
- Deployment of a database ingestion and replication task that has an SAP HANA source and Microsoft Azure Synapse Analytics target might fail if a source table contains a multiple-column primary key of a long length. In this case, reduce the length of the primary key and then deploy the task again.

Teradata sources

To use Teradata sources in database ingestion and replication tasks, first prepare the source database and review the usage considerations.

Source preparation

- To deploy and run a database ingestion and replication task that includes a Teradata source, the source connection must specify a database user who has the privileges that are required to perform an initial load operation. Use the following SQL statements to grant these privileges to the user:

```
GRANT SELECT ON database_name TO user_name/user_role;
```

Usage considerations

- Database ingestion and replication jobs that have a Teradata source can have any target type except for PostgreSQL and SQL Server.
- Database Ingestion and Replication does not support the following Teradata data types:
  - ARRAY
  - BLOB
  - CLOB
  - JSON
  - ST_GEOMETRY
  - XML

  For information about the default mappings of supported source data types to target data types, see "Default Data Type Mappings" on page 172.

Database Ingestion and Replication targets - preparation and usage

Before you configure database ingestion and replication tasks for initial load, incremental load, or combined initial and incremental load operations, review the following guidelines for your target types to avoid unexpected results:

Amazon Redshift targets

The following list identifies considerations for preparing and using Amazon Redshift targets:

- You can use either Amazon Redshift or Amazon Redshift Serverless targets in database ingestion and replication jobs.
- Before writing data to Amazon Redshift target tables, database ingestion and replication jobs stage the data in an Amazon S3 bucket. You must specify the name of the bucket when you configure the database ingestion and replication task. The database ingestion and replication jobs use the COPY command to load the data from the Amazon S3 bucket to the Amazon Redshift target tables. For more information about the COPY command, see the Amazon Web Services documentation.
- When you define a connection for an Amazon Redshift target, provide the access key and secret access key for the Amazon S3 bucket in which you want the database ingestion and replication jobs to stage the data before loading it to the Amazon Redshift target tables.
- Incremental load jobs and combined initial and incremental load jobs generate a recovery table named INFORMATICA_CDC_RECOVERY on the target to store internal service information. The data in the recovery table prevents jobs that are restarted after a failure from propagating previously processed data again. The recovery table is generated in the same schema as the target tables.
- Database ingestion and replication jobs convert infinity and NaN values in FLOAT columns in source tables to null values in Amazon Redshift target tables.

Amazon S3, Flat File, Google Cloud Storage, Microsoft Azure Data Lake Storage, Microsoft Fabric OneLake, and Oracle Cloud Object Storage targets

The following list identifies considerations for using Amazon S3, Flat File, Google Cloud Storage, Microsoft Azure Data Lake Storage, Microsoft Fabric OneLake, and Oracle Cloud Infrastructure (OCI) Object Storage targets:

- When you define a database ingestion and replication task that has an Amazon S3, Google Cloud Storage, Microsoft Azure Data Lake Storage, Microsoft Fabric OneLake, or Oracle Cloud Object Storage target, you can select CSV, Avro, or Parquet as the format for the generated output files that contain the source data to be applied to the target. For flat file targets, you can select either CSV or Avro as the output file format.
- If you select the CSV output format, Database Ingestion and Replication creates the following files on the target for each source table:
  - A schema.ini file that describes the schema and includes some settings for the output file on the target.
  - One or multiple output files for each source table, which contain the source data. Database Ingestion and Replication names these text files based on the name of the source table with an appended date and time.

  The schema.ini file lists a sequence of columns for the rows in the corresponding output file. The following table describes the columns in the schema.ini file:

Column	Description
ColNameHeader	Indicates whether the source data files include column headers.
Format	Describes the format of the output files. Database Ingestion and Replication uses a comma (,) to delimit column values.

Column	Description
CharacterSet	Specifies the character set that is used for output files. Database Ingestion and Replication generates the files in the UTF-8 character set.
COL*<sequence_number>*	The name and data type of the column. **Notes:** - If you selected any of the **Add Operation...** properties under **Advanced** on the **Target** page of the task wizard, the list of columns includes metadata columns for the operation type, time, owner, or transaction ID. - If you selected the **Add Before Images** check box, for each source column, the job creates a *column_name*_OLD column for UNDO data and *column_name*_NEW column for REDO data.

Important: You should not edit the schema.ini file.

- If you select the Avro output format, you can select an Avro format type, a file compression type, an Avro data compression type, and the directory that stores the Avro schema definitions generated for each source table. The schema definition files have the following naming pattern: *schemaname_tablename*.txt.
- If you select the Parquet output format, you can optionally select a compression type that Parquet supports.
- On Flat File, Microsoft Azure Data Lake Storage, and Microsoft Fabric OneLake targets, Database Ingestion and Replication creates an empty directory for each empty source table. Database Ingestion and Replication does not create empty directories on Amazon S3, Google Cloud Storage, and Oracle Cloud Object Storage targets.
- If you do not specify an access key and secret key in the Amazon S3 connection properties, Database Ingestion and Replication tries to find AWS credentials by using the default credential provider chain that is implemented by the DefaultAWSCredentialsProviderChain class. For more information, see the *Amazon Web Services* documentation.
- If database ingestion and replication incremental load and combined initial and incremental load jobs replicate Update operations that change primary key values on the source to any of these targets that use the CSV output format, the job processes each Update record as two records on the target: a Delete followed by an Insert. The Delete contains the before image. The Insert contains the after image for the same row.

  For Update operations that do not change primary key values, database ingestion and replication jobs process each Update as one operation and writes only the after image to the target.

  Note: If source tables do not have primary keys, Database Ingestion and Replication treats the tables as if all columns were part of the primary key. In this case, each Update operation is processed as a Delete followed by an Insert.
- Database Ingestion and Replication jobs unload binary data in hexadecimal format when the data is sent to an Amazon S3, Flat File, Microsoft Azure Data Lake Storage, or Microsoft Fabric OneLake target. Each hexadecimal column value has the "0x" prefix. If you want to use output files to load the data to a target, you might need to edit the files to remove the "0x" prefixes.
- If you run a Secure Agent service on Windows and plan to use Flat File connections, ensure that the logon account for the Secure Agent is an Administrator account. Otherwise, an error occurs when you try to configure a Flat File connection.

Databricks targets

To use Databricks targets in database ingestion and replication tasks, first prepare the target and review the usage considerations.

Target preparation:

1. Download the DatabricksJDBC42 JDBC driver version 2.6.25 from the Databricks JDBC driver downloads website.

   If you want to use the same driver for both Data Ingestion and Replication and Cloud Data Integration, note that Data Integration provides the 2.6.22 version of the driver in its connector package. In this case, copy the DatabricksJDBC42.jar file from the Data Integration connector package instead of downloading it from the Databricks website.

   Note: Data Ingestion and Replication can use version 2.6.5 or later or version 2.6.22. If you use version 2.6.5 or later, the **Test Connection** feature in the Administrator interface does not work.

2. Copy the DatabricksJDBC42.jar file to the following directory:

   Secure_Agent_installation_directory/apps/Database_Ingestion/ext/

   If the older SparkJDBC42.jar file is already installed in the Secure Agent installation directory, remove it.

3. In the Databricks connection properties, set the **JDBC Driver Class Name** property to `com.databricks.client.jdbc.Driver`.

   Note: If you downloaded the older Cloud Data Integration driver, specify `com.simba.spark.jdbc.Driver` as the class name.

4. On Windows, install Visual C++ Redistributable Packages for Visual Studio 2013 on the computer where the Secure Agent runs.

Usage considerations:

- For incremental load jobs, you must enable Change Data Capture (CDC) for all source columns.
- You can access Databricks tables created on top of the following storage types:

  - Microsoft Azure Data Lake Storage (ADLS) Gen2

  - Amazon Web Services (AWS) S3

  The Databricks connection uses a JDBC URL to connect to the Databricks cluster. When you configure the target, specify the JDBC URL and credentials to use for connecting to the cluster. Also define the connection information that the target uses to connect to the staging location in Amazon S3 or ADLS Gen2.

- Before writing data to Databricks target tables, database ingestion and replication jobs stage the data in an Amazon S3 bucket or ADLS directory. You must specify the directory for the data when you configure the database ingestion and replication task.

  Note: Database Ingestion and Replication does not use the **ADLS Staging Filesystem Name** and **S3 Staging Bucket** properties in the Databricks connection properties to determine the directory.

- Database Ingestion and Replication uses jobs that run once to load data from staging files on Amazon S3 or Amazon Data Lake Store Gen2 to external tables.
  By default, Database Ingestion and Replication runs jobs on the cluster that is specified in the Databricks connection properties. If you want to run jobs on another cluster, set the dbDeltaUseExistingCluster custom property to false on the **Target** page in the database ingestion and replication task wizard.
- If the cluster specified in the Databricks connection properties is not up and running, the database ingestion and replication job waits for the cluster to start. By default, the job waits for 10 minutes. If the cluster does not start within 10 minutes, the connection times out and deployment of the job fails.
  If you want to increase the timeout value for the connection, set the dbClusterStartWaitingTime custom property to the maximum time in seconds for which the ingestion job must wait for the cluster to be up

and running. You can set the custom property on the **Target** page in the database ingestion and replication task wizard.

- By default, Database Ingestion and Replication uses the Databricks COPY INTO feature to load data from the staging file to Databricks target tables. You can disable it for all load types by setting the writerDatabricksUseSqlLoad custom property to false on the **Target** page in the database ingestion and replication task wizard.
- If you use an AWS cluster, you must specify the **S3 Service Regional Endpoint** value in the Databricks connection properties. For example:

  ```
  s3.us-east-2.amazonaws.com
  ```

  To test a Databricks connection using a Secure Agent on Linux, you must specify the JDBC URL in the **SQL Endpoint JDBC URL** field in the Databricks connection properties. After you test the connection, remove the **SQL Endpoint JDBC URL** value. Otherwise, when you define a database ingestion and replication task that uses the connection, a design-time error occurs because Data Ingestion and Replication tries to use the JDBC URL as well as the required **Databricks Host**, **Cluster ID**, **Organization ID**, and **Databricks Token** values to connect to target, resulting in login failures.
- You cannot test a Databricks connection using a Secure Agent on Windows. The test fails. In this situation, you can perform the test using a Secure Agent on Linux. However, note that you can use a Databricks connection with a Secure Agent on Windows when creating a database ingestion and replication task or running a database ingestion and replication job.
- Processing of Rename Column operations on Databricks target tables, without the need to rewrite the underlying Parquet files, requires the Databricks Column Mapping feature with Databricks Runtime 10.2 or later. If you set the **Rename Column** option to **Replicate** on the **Schedule and Runtime Options** page in the task wizard, you must alter the generated target table to set the following Databricks table properties after task deployment and before you run the job:

  ```
  ALTER TABLE <target_table> SET TBLPROPERTIES (
  'delta.columnMapping.mode' = 'name',
  'delta.minReaderVersion' = '2',
  'delta.minWriterVersion' = '5')
  ```

  These properties enable the Databricks Column Mapping feature with the required reader and writer versions. If you do not set these properties, the database ingestion and replication job will fail.
- Database ingestion and replication jobs that have Databricks targets can get schema information for generating the target tables from Databricks Unity Catalog. To enable access to information in a Unity Catalog, specify the catalog name in the **Catalog Name** field in the Databricks connection properties. The catalog name is appended to the **SQL Warehouse JDBC URL** value for a data warehouse.

  Note: Catalog use is optional for SQL warehouses and does not apply to job clusters.

  If you use Unity Catalog, a personal storage location is automatically provisioned. To use the personal staging location, in the **Staging Environment** field of the connection properties, select **Personal Staging Location**. The Parquet data files for ingestion and replication jobs can then be staged to the local personal storage location, which has a data retention period of 7 days. By default, the staging location is `stage://tmp/<user_name>` in the root AWS or Azure location. The <user_name> is taken from the **Database Token** connection property. This user must have read and write access to the personal staging location.
- Database Ingestion and Replication supports special characters in table and column names in the source tables. The special characters are replaced with an underscore (_) in the Databricks target table or column names.
  The custom property key-value pair `targetReplacementValue=toHex` prevents Database Ingestion and Replication from replacing the special characters with an underscore in the generated target schema and converts the special characters to hexadecimal format.

To convert the special characters to their hexadecimal value, perform the following actions before deploying the database ingestion and replication task:

1. Create a properties file to be used at the metadata-manager layer. Add the `targetReplacementValue=toHex` key-value pair to the properties file.
2. Open the **Runtime Environment** page in Administrator and edit the Secure Agent. Create a custom property in the **Custom Configuration Details** area:
   - Select the **Database Ingestion** service.
   - Select the **DBMI_AGENT_ENV** type.
   - Enter DBMI_TASK_OVERRIDE_PROPERTIES as the property name.
   - Enter the properties file location as the property value.
3. Set the targetReplacementValue custom property to toHex on the **Target** page of the task wizard.

   Before running the task, add <jobname>. to the targetReplacementValue key in the properties file:

   ```
   <jobname>.targetReplacementValue=toHex
   ```

   If the property affects all jobs, add "alljobs." to the targetReplacementValue key:

   ```
   alljobs.targetReplacementValue=toHex
   ```

- If you select source decimal or numeric columns to replicate to Databricks target columns, ensure that each source decimal or numeric column has a scale that is less than or equal to its precision. Otherwise, when you run the job, an error is issued that reports an invalid decimal scale. This consideration applies to any source type that is mapped to a Databricks target.
- If you generated Databricks unmanaged tables as the target tables and no longer need a target unmanaged table, use a SQL DROP TABLE statement to delete the table from the target database. You must not manually delete the external directory in Amazon S3 or Azure Data Lake Storage for the unmanaged table. If you do so and try to deploy another job that uses that table, the deployment fails with a Metadata Handler error.

Google BigQuery targets

The following list identifies considerations for preparing and using Google BigQuery targets:

Target preparation

- Download and install the Google BigQuery JDBC driver.
  1. Download the Google BigQuery JDBC driver version 1.2.25.1029 from the Google Cloud website.
  2. Copy all of the jar files in the installation zip to the following directory:

     ```
     Secure Agent installation directory/apps/Database_Ingestion/ext/
     ```

  3. Restart the Secure Agent.
- Ensure that you have a service account in your Google account to access Google BigQuery and Google Cloud Storage.
- Ensure that you have the client_email, project_id, private_key, and region ID values for the service account. Enter the values in the corresponding **Service Account ID**, **Project ID**, **Service Account Key**, and **Region ID** connection properties when you create a Google BigQuery connection.
- If you want to configure a timeout interval for a Google BigQuery connection, specify the timeout interval property in the **Provide Optional Properties** field of the connection properties. Use the following format:

  ```
  "timeout": "<timeout_interval_in_seconds>"
  ```

- Verify that you have read and write access to the following entities:

  - Google BigQuery datasets that contains the target tables

- Google Cloud Storage path where Database Ingestion and Replication creates the staging file

- To use a Google BigQuery target, you must configure the required permissions. First, create an IAM & Admin service account in your Google Cloud project and assign the Custom Role to it. Then add the following permissions to the account's custom role:
  - bigquery.datasets.get - To get metadata about a data set.
  - bigquery.jobs.create - To run jobs and queries.
  - bigquery.models.create - To create new models.
  - bigquery .models.delete - To delete models.
  - bigquery .models.export - To export a model.
  - bigquery.models.getData - To get model data. The bigquery.models.getMetadata permission must also be specified.
  - bigquery.models.getMetadata - To get model metadata. The bigquery.models.getData permission must also be specified.
  - bigquery .models.list - To list models and metadata for the models.
  - bigquery.models.updateData - To update model data. The bigquery.models.updateMetadata permission must also be specified.
  - bigquery.models.updateMetadata - To update model metadata. The bigquery.models.updateData permission must also be specified.
  - bigquery.routines.create - To create new routines, including stored procedures.
  - bigquery.routines.delete - To delete routines.
  - bigquery.routines.get - To get routine definitions and metadata.
  - bigquery.routines.list - To list routines and metadata for the routines.
  - bigquery.routines.update - To update routine definitions and metadata for the routines.
  - bigquery.routines.updateTag - To update tags for routines.
  - bigquery.tables.create - To create new tables.
  - bigquery.tables.delete - To delete tables.
  - bigquery.tables.deleteIndex - To drop search indexes on tables.
  - bigquery.tables.deleteSnapshot - To delete table snapshots.
  - bigquery.tables.export - To export table data out of BigQuery.
  - bigquery.tables.get - To get table metadata. The bigquery.tables.getData permission must also be specified.
  - bigquery.tables.getData - To get and query table data. The bigquery.tables.get permission must also be specified.
  - bigquery.tables.list - To list tables and metadata for the tables.
  - bigquery.tables.update - To update table metadata. The bigquery.tables.updateData permission must also be specified.
  - bigquery.tables.updateData - To update table data. Thebigquery.tables.update permission must also be specified.
  - bigquery.tables.updateTag - To update tags for tables.
  - resourcemanager.projects.get - Get the name of the billing account associated with the project.
  - storage.objects.create - To allow users to create objects.

- storage.objects.delete - To grant permissions to delete objects.
- storage.objects.get - To read object metadata when listing and reading bucket metadata.
- storage.objects.list - To list objects in a bucket.

Target usage

- Database Ingestion and Replication loads source data in bulk mode to Google BigQuery targets.
- For database ingestion and replication incremental load tasks, make sure that source database Change Data Capture (CDC) is enabled on all source columns.
- Database Ingestion and Replication supports special characters in table and column names in the source tables. The special characters are replaced with an underscore (_) in the Google BigQuery target table or column names.
  The custom property key-value pair `targetReplacementValue=toHex` prevents Database Ingestion and Replication from replacing the special characters with an underscore in the generated target schema and converts the special characters to hexadecimal format.

  To convert the special characters to their hexadecimal value, perform the following actions before deploying the database ingestion and replication task:

  1. Create a properties file to be used at the metadata-manager layer. Add the `targetReplacementValue=toHex` key-value pair to the properties file.
  2. Open the **Runtime Environment** page in Administrator and edit the Secure Agent. Create a custom property in the **Custom Configuration Details** area:
     - Select the **Database Ingestion** service.
     - Select the **DBMI_AGENT_ENV** type.
     - Enter DBMI_TASK_OVERRIDE_PROPERTIES as the property name.
     - Enter the properties file location as the property value.
  3. Set the targetReplacementValue custom property to toHex on the **Target** page of the task wizard.

  Before running the task, add <jobname>. to the "targetReplacementValue" key in the properties file:

  ```
  <jobname>.targetReplacementValue=toHex
  ```

  If the property affects all jobs, add "alljobs." to the "targetReplacementValue" key:

  ```
  alljobs.targetReplacementValue=toHex
  ```
- If you select **Audit** apply mode for a database ingestion and replication task, you can select the audit metadata columns to include in the target tables on the task wizard's **Target** page under **Advanced**. If you specify a value in the **Prefix for Metadata Columns** field, do not include special characters. Otherwise, task deployment will fail.
- By default, the following types of source columns are mapped to Google BigQuery string columns with no length specification:
  - Source columns that have a character data type
  - SAP HANA source columns that have the longdate or timestamp data type
- When you deploy database ingestion and replication jobs, Database Ingestion and Replication generates Google BigQuery target tables clustered by primary key or unique key columns, by default. Each key column must have one of the following data types that Google BigQuery supports for clustering:
  - STRING
  - INT64
  - NUMERIC
  - BIGNUMERIC

- DATE
- DATETIME
- TIMESTAMP
- BOOL
- GEOGRAPHY

If any column in the primary key or unique key has an unsupported data type, that column is skipped during clustering. For example, if the primary key contains the C1, C2, C3, C4, C5 columns and C2 has an unsupported data type, the target table is created with the C1, C3, C4, and C5 columns in the CLUSTER BY clause.

Kafka targets and Kafka-enabled Azure Event Hubs targets

The following list identifies considerations for using Kafka targets:

- Database Ingestion and Replication supports Apache Kafka, Confluent Kafka, Amazon Managed Streaming for Apache Kafka (MSK), and Kafka-enabled Azure Event Hubs as targets for incremental load jobs. All of these Kafka target types use the Kafka connection type.

  To indicate the Kafka target type, you must specify Kafka producer properties in the task definition or Kafka connection properties. To specify these properties for a task, enter a comma-separated list of *key*:*value* pairs in the **Producer Configuration Properties** field on the **Target** page of the task wizard. To specify the producer properties for all tasks that use a Kafka connection, enter the list of properties in the **Additional Connection Properties** field in the connection properties. You can override the connection-level properties for specific tasks by also defining producer properties at the task level. For more information about producer properties, see the Apache Kafka, Confluent Kafka, Amazon MSK, or Azure Event Hubs for Kafka documentation.

- If you select **AVRO** as the output format for a Kafka target, Database Ingestion and Replication generates a schema definition file for each table with a name in the following format:

  ```
  schemaname_tablename.txt
  ```

  If a source schema change is expected to alter the target in an incremental load job, Database Ingestion and Replication regenerates the Avro schema definition file with a unique name that includes a timestamp:

  ```
  schemaname_tablename_YYYYMMDDhhmmss.txt
  ```

  This unique naming pattern preserves older schema definition files for audit purposes.

- If you have a Confluent Kafka target that uses Confluent Schema Registry to store schemas, you must configure the following settings on the **Target** page of the task wizard:

  - In the **Output Format** field, select **AVRO**.
  - In the **Avro Serialization Format** field, select **None**.

- You can specify Kafka producer properties in either the **Producer Configuration Properties** field on the **Target** page of the task wizard or in the **Additional Connection Properties** field in the Kafka connection properties. Enter property=value pairs that meet your business needs and that are supported by your Kafka vendor.

  For example, if you use Confluent Kafka, you can use the following entry in either the **Producer Configuration Properties** field or **Additional Connection Properties** field to specify the Schema Registry URL and enable basic authentication:

  ```
  schema.registry.url=http://schema-registry:8081,
  key.serializer=org.apache.kafka.common.serialization.StringSerializer,
  value.serializer=io.confluent.kafka.serializers.KafkaAvroSerializer,
  ```

```
basic.auth.credentials.source=USER_INFO,
basic.auth.user.info=myname:mypassword
```

If you use Amazon MSK, you can use the following **Additional Connection Properties** entry to enable IAM role authentication for access to Amazon MSK targets:

```
security.protocol=SASL_SSL,sasl.mechanism=AWS_MSK_IAM,sasl.jaas.config=software.amazon
.msk.auth.iam.IAMLoginModule
required;,sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCal
lbackHandler
```

Ensure that you enable IAM role authentication on the Amazon EC2 instance where the Secure Agent is installed.

For more information about Kafka properties, see the documentation of your Kafka vendor.

- Database ingestion and replication incremental load jobs can replicate change data to Kafka targets that support SASL_SSL secured access, including Confluent Kafka, Amazon MSK, and Azure Event Hubs targets. In Administrator, you must configure a Kafka connection that includes the appropriate properties in the **Additional Connection Properties** field. For example, for Azure Event Hubs, you could use the following **Additional Connection Properties** entry to enable SASL_SSL:

```
bootstrap.servers=NAMESPACENAME.servicebus.windows.net:9093
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required
username="$ConnectionString" password="{YOUR.EVENTHUBS.CONNECTION.STRING}";
```

Generating custom message keys for Kafka targets

For all Kafka target types that use the Avro format, you can configure rules to generate a custom message key that consists of one or more columns for each source table. Database ingestion and replication incremental load jobs that have a Kafka target can then include the generated message key in the headers of the messages that it sends to the target messaging system. The target messaging system can use the message key to write messages with a specific key value to the same partition in a multi-partitioned topic.

To implement this feature, you must manually create a configuration file that contains rules that identify the key columns for each source table. Then specify the file in a custom configuration property in the task wizard.

Configuration file creation

Create the rule-configuration file in a text editor and save it to a location on the Secure Agent system. The file contains a rule for each source table. Each rule defines the custom key column or columns to use for writing data to topic partitions.

Note: If you change or add a rule or change one of the other parameters after the database ingestion and replication task has been deployed, you must redeploy the task for the rule change to take effect.

Rule syntax:

Use the following syntax to define rules in the configuration file:

```
rule=(schema.tablename,column1,column2,column3,… )
additional rules...
[tableNotFound=ABORT]
[trace={true|false}]
[delimiter=character]
```

To include comments in the file, begin each comment line with the number (#) sign. For example:

```
#This text is for informational purposes only.
```

Parameters:

- **rule**. Defines a rule for generating a composite message key for a source table. In each rule, first identify the schema and table name for a source table. If you change the schema or define table renaming rules for the target, use the name of the schema or renamed table on the target. Then specify the names of one or more table columns that comprise the message key. Ensure that the columns are defined in the table. Otherwise, the database ingestion job fails. For SQL Server sources, also include the name of the database in the format: *database*.*schema*.*tablename*.

  You can define multiple rules in the same rule-configuration file.

  When generating the message key, Data Ingestion and Replication uses the character representation of each column value followed by the delimiter character. Each column value and delimiter are appended to the composite key value in the order in which the columns appear in the rule definition. The composite key is then used as the Kafka message key for a record. The position of any column that has empty or null values in the message key is represented by the delimiter character only.

- **delimiter**. *Optional*. Specifies a single character that will be used as the delimiter after each key column value in generated message keys. You can specify this parameter only once in the rule-configuration file.

  Default is the semicolon (;).

- **tableNotFound**. *Optional*. Set this parameter to ABORT to cause database ingestion and replication jobs to stop processing any data for a source table and then fail when the table does not have a rule definition in the rule-configuration file. Each source table must have a rule definition for the generation of the composite message key to succeed. You can specify this parameter only once in the configuration file.

  If you do not specify this parameter and the table is not found in the rule-configuration file, the default rule in the target messaging system parameters determines the key to use for a record.

- **trace**. *Optional*. Enables or disables tracing for the generation of message keys based on rule definitions. Valid values are:

  - **true**. Enables tracing for message key generation based on rule definitions.

  - **false**. Disables tracing for message key generation based on rule definitions.

  You can specify this parameter only once in the rule-configuration file.

  Default is **false**.

Example rule:

```
rule=(testdb.ABC.DEPT,DEPTNO,DNAME)
tableNotFound=ABORT
trace=true
delimiter=;
```

Example generated key output based on this rule:

```
1234;HR;
```

Database ingestion and replication task configuration

When you create a database ingestion and replication incremental load task that has a Kafka target, you must set the following options to enable the generation of custom message keys:

- On the **Target** page of the task wizard, make sure that the **Use Table Name as Topic Name** check box is cleared. Then enter the topic name in the **Topic Name** field.
- In the **Output Format** field, select **Avro**. You can select any Avro format in the **Avro Format** field.
- Under **Custom Properties**, specify the **captureColumnValuesFile** property with a path value that points to the rule-configuration file you created on the Secure Agent system.

Microsoft Azure Synapse Analytics targets

The following list identifies considerations for preparing and using Microsoft Azure Synapse Analytics targets:

- To deploy and run a database ingestion and replication task with a Microsoft Azure Synapse Analytics target, the target connection must specify a database user who has the CONTROL permission on the target database. To grant the CONTROL permission to a user, use the following SQL statements:

  ```
  USE database_name;
  GRANT CONTROL TO user_name;
  ```

  This permission is required for initial load, incremental load, and combined initial and incremental load jobs. This permission allows Database Ingestion and Replication to create target tables and database objects, such as external data source, external file format, and database scoped credential objects, if they do not exist in the database. This permission is specifically required for creating external data source and database scoped credential objects.

  Note: You must manually create the master key. To create the master key, you must have the CONTROL permission on the database.

- Database ingestion and replication jobs first send data to a Microsoft Azure Data Lake Storage Gen2 staging file before writing the data to Microsoft Azure Synapse Analytics target tables. The staging file uses the hexadecimal x1d separator as the field delimiter. After the data is written to the target, the entire contents of the table-specific directory that includes the staging files are deleted.
- If you use Microsoft Azure Data Lake Storage Gen2 with a Synapse Analytics connection, you must enable the **Hierarchical namespace** option in Microsoft Azure Data Lake Storage. With this setting, use of blob storage is not recommended.
- The number of columns in a source table that a database ingestion and replication job can propagate to a Microsoft Azure Synapse Analytics target must not exceed 508 columns.
- Database ingestion and replication incremental load jobs and combined initial and incremental load jobs generate a recovery table named INFORMATICA_CDC_RECOVERY on the target to store internal service information that prevents jobs restarted after a failure from propagating previously processed data again. This recovery table is generated in the same schema as the target tables.
- After a database ingestion and replication job loads data to a Microsoft Azure Synapse Analytics target by using external tables, the job does not drop the log tables and external tables created on the target, even though these tables might be re-created when the job starts again.

Microsoft SQL Server and Azure SQL Database targets

You can use Microsoft SQL Server or Microsoft Azure SQL Database targets in initial load, incremental load, and combined initial and incremental load jobs. SQL Server targets include on-premises, RDS, and Azure SQL Managed Instance targets.

The following list identifies considerations for preparing and using Microsoft SQL Server targets:

- Database Ingestion and Replication does not support Microsoft SQL Server as a target in jobs that replicate data from Teradata sources.
- The SQL Server JDBC driver is delivered with Database Ingestion and Replication. You do not need to install it separately.
- The Database Ingestion and Replication user requires following database roles, at minimum, to create target tables and write data to the tables:

  - db_datareader

  - db_datawriter

- db_ddladmin

- In Administrator, when you define a SQL Server connection for connecting to a SQL Server target, set the following required connection properties:

  - SQL Server Version. Select either **SQL Server 2017** or **SQL Server 2019**.

  - Authentication Mode. Select **SQL Server Authentication** or **Windows Authentication v2**.

  - User Name

  - Password

  - Host

  - Port

  - Database Name

  - Schema

  - Code Page

  Optionally, to use SSL encryption for the target connection, you can set the Encryption Method property to one of the following options: SSL, Request SSL, or Login SSL. If you enable encryption, also set the following properties:

  - Crypto Protocol Version

  - Validate Server Certificate. Select **True**.

  - Trust Store

  - Trust Store Password

  - Host Name in Certificate

  Other connection properties are not supported.

- Database ingestion and replication incremental load and initial and incremental load jobs with a SQL Server target generate a LOG table based on the target table schema, with some additional metadata columns. The LOG table is created right before change data is flushed to the target. The incoming DML data is inserted to the LOG table by supplying a local CSV file to the Bulk Copy API of the SQL Server driver. A merge apply statement is generated based on the information in the LOG table, and then the DML operations are applied to the actual target table. After the DML changes are applied, the LOG table is dropped.
  The LOG table might cause a temporary spike in additional space or size requirements in the customer database instance if you run multiple jobs or a job with multiple tables. The space and size required by the LOG table depends on the number of rows received as part of a flush cycle.

- The number of columns in a source table that a database ingestion and replication incremental load or an initial and incremental load job can propagate to a SQL Server target must not exceed 508 columns. If a source table contains more than 508 columns, the job fails while creating the LOG table.

- Database ingestion and replication incremental load jobs and combined initial and incremental load jobs generate a recovery table named INFORMATICA_CDC_RECOVERY on the target to store internal service information that prevents jobs restarted after a failure from propagating previously processed data again. This recovery table is generated in the same schema as the target tables.

Oracle targets

The following list identifies considerations for preparing and using Oracle targets.

Target preparation

- Database Ingestion and Replication requires users to have certain privileges to load data to Oracle target databases. For on-premises Oracle targets, grant the following user privileges to the Database Ingestion and Replication user (*cmid_user*) who connects to the Oracle target:

```
GRANT CREATE SESSION TO cmid_user;

GRANT SELECT ON "PUBLIC".V$DATABASE TO cmid_user;
GRANT SELECT ON "PUBLIC".V$CONTAINERS TO cmid_user;

GRANT SELECT ON DBA_USERS TO cmid_user;
GRANT SELECT ON DBA_TABLES TO cmid_user;
GRANT SELECT ON DBA_OBJECT_TABLES TO cmid_user;
GRANT SELECT ON DBA_INDEXES TO cmid_user;
GRANT SELECT ON DBA_OBJECTS TO cmid_user;
GRANT SELECT ON DBA_VIEWS TO cmid_user;

GRANT CREATE TABLE <schema.table> TO cmid_user; <--Unless you grant on ANY TABLE
GRANT SELECT ON ALL_CONSTRAINTS TO cmid_user;
GRANT SELECT ON ALL_OBJECTS TO cmid_user;

GRANT SELECT ON SYS.TAB$ TO cmid_user;
GRANT SELECT ON SYS.RECYCLEBIN$ TO cmid_user;
GRANT SELECT ON SYS.COL$ TO cmid_user; <-- If cmid_user is the owner of the target
schema
GRANT SELECT ON SYS.CCOL$ TO <cmid_user>;
GRANT SELECT ON SYS.CDEF$ TO cmid_user;
GRANT SELECT ON SYS.OBJ$ TO cmid_user;
GRANT SELECT ON SYS.COLTYPE$ TO cmid_user;
GRANT SELECT ON SYS.ATTRCOL$ TO cmid_user;
GRANT SELECT ON SYS.IDNSEQ$ TO cmid_user;
GRANT SELECT ON SYS.ATTRCOL$ TO cmid_user;
GRANT SELECT ON SYS.IDNSEQ$ TO cmid_user;
GRANT SELECT ON SYS.IND$ TO cmid_user;

-- Grant the following if cmid_user is NOT the owner of the target schema. If you
prefer, you
-- can grant to individual target tables and indexes instead of to ANY TABLE or ANY
INDEX.
GRANT ALTER SESSION TO cmid_user;
GRANT RESOURCE TO cmid_user;
GRANT SELECT ANY TABLE TO cmid_user;
GRANT SELECT ANY DICTIONARY TO <cmid_user>;
GRANT ALTER ANY TABLE TO cmid_user;
GRANT CREATE ANY TABLE TO cmid_user;
GRANT DROP ANY TABLE TO cmid_user;
GRANT INSERT ANY TABLE TO cmid_user;
GRANT UPDATE ANY TABLE TO cmid_user;
GRANT DELETE ANY TABLE TO cmid_user;
GRANT CREATE ANY INDEX TO cmid_user;
GRANT ALTER ANY INDEX TO cmid_user;
GRANT DROP ANY INDEX TO cmid_user;
```

- For Amazon RDS for Oracle targets, log in to RDS as the master user and grant the following user privileges to the Database Ingestion and Replication user (*cmid_user*) who connects to the Oracle target:

```
GRANT CREATE SESSION TO cmid_user;

GRANT SELECT on "PUBLIC".V$DATABASE TO cmid_user;

GRANT SELECT on DBA_USERS TO cmid_user;
GRANT SELECT on DBA_TABLES TO cmid_user;
GRANT SELECT on DBA_INDEXES TO cmid_user;
GRANT SELECT ON DBA_VIEWS TO cmid_user;

GRANT CREATE TABLE <schema.table> TO cmid_user;
GRANT SELECT on SYS.TAB$ TO cmid_user;
GRANT SELECT on SYS.COL$ TO cmid_user;
GRANT SELECT on SYS.OBJ$ TO cmid_user;
```

```
GRANT SELECT on SYS.IND$ TO cmid_user;

-- Grant the following if cmid_user is NOT the owner of the target schema. If you
prefer, you
-- can grant to individual target tables and indexes instead of to ANY TABLE or INDEX.
GRANT ALTER SESSION TO cmid_user;
GRANT RESOURCE TO cmid_user;
GRANT SELECT ANY TABLE TO cmid_user;
GRANT SELECT ANY DICTIONARY TO <cmid_user>;
GRANT ALTER ANY TABLE TO cmid_user;
GRANT CREATE ANY TABLE TO cmid_user;
GRANT DROP ANY TABLE TO cmid_user;
GRANT INSERT ANY TABLE TO cmid_user;
GRANT UPDATE ANY TABLE TO cmid_user;
GRANT DELETE ANY TABLE TO cmid_user;
GRANT CREATE ANY INDEX TO cmid_user;
GRANT ALTER ANY INDEX TO cmid_user;
GRANT DROP ANY INDEX TO cmid_user;
```

Also, run the following Amazon RDS procedures to grant additional SELECT privileges to *cmid_user*:

```
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'V_$CONTAINERS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_OBJECT_TABLES',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'DBA_OBJECTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_CONSTRAINTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ALL_OBJECTS',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'RECYCLEBIN$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT');
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'CCOL$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
```

```
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'CDEF$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'COLTYPE$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'ATTRCOL$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
begin
rdsadmin.rdsadmin_util.grant_sys_object(
p_obj_name => 'IDNSEQ$',
p_grantee => 'cmid_user',
p_privilege => 'SELECT',
p_grant_option => false);
end;
/
```

Usage considerations

- By default, Database Ingestion and Replication disables logging for the Oracle target table to optimize performance. You can enable logging by setting the writerOracleNoLogging custom property to false on the **Target** page in the database ingestion and replication task wizard.
- Database ingestion and replication initial load jobs that have an Oracle target fail if SSL use is enabled in the Oracle Database Ingestion connection properties.

PostgreSQL targets

You can use Amazon Aurora PostgreSQL and RDS for PostgreSQL as a target in initial load, incremental load, and combined initial and incremental load jobs that have a Db2 for i or Oracle source.

The following list identifies considerations for preparing and using PostgreSQL targets:

- To deploy and run a database ingestion and replication task that includes a PostgreSQL target, the target connection must specify a database user who has the required privileges.
  The user must have the CONNECT and TEMPORARY privileges for the database specified in the connection and the USAGE and CREATE privileges for the target schema specified in the target properties.

  Use the following SQL statements to grant these privileges to a user role and then assign the role to a user:

  ```
  CREATE ROLE dbmi_role;
  GRANT CONNECT ON DATABASE database TO dbmi_role;
  GRANT TEMPORARY ON DATABASE database TO dbmi_role;
  GRANT CREATE ON SCHEMA schema TO dbmi_role;
  GRANT USAGE ON SCHEMA schema TO dbmi_role;
  CREATE USER dbmi_user with PASSWORD 'password';
  GRANT dbmi_role to dbmi_user;
  ```

Note: Privileges on the target tables that are generated when the job runs are granted to the user who runs the job.

- Database ingestion and replication incremental load jobs with a PostgreSQL target generate a LOG table based on the target table schema, with some additional metadata columns. The LOG table is created right before change data is flushed to the target. The incoming DML data is inserted to the LOG table by supplying a local CSV file to the Bulk Copy API of the PostgreSQL driver. A set of merge apply statements is generated based on the information in the LOG table, and then the DML operations are applied to the actual target table. After the DML changes are applied, the LOG table is dropped.
  The LOG table might cause a temporary spike in additional space or size requirements in the customer database instance if you run multiple jobs or a job with multiple tables. The space and size required by the LOG table depends on the number of rows received as part of a flush cycle.
- The number of columns in a source table that a database ingestion and replication incremental load job can propagate to a PostgreSQL target must not exceed 796 columns. If a source table contains more than 796 columns, the job fails while creating the LOG table.
- PostgreSQL supports a maximum length of 63 characters for source object identifiers. If the length of any source table or column name exceeds 63 characters, the deployment of database ingestion and replication tasks with a PostgreSQL target will fail during validation.
- Database ingestion and replication incremental load jobs generate a recovery table named INFORMATICA_CDC_RECOVERY on the target to store internal service information that prevents jobs restarted after a failure from propagating previously processed data again. This recovery table is generated in the same schema as the target tables.

Snowflake targets

Target preparation

Target preparation varies depending on whether you use the Superpipe feature for high performance streaming of data to Snowflake target tables or write data to intermediate stage files.

With Superpipe

If you use the Superpipe feature, complete the following steps:

1. Create a Data Ingestion and Replication user. Use the following SQL statement:

   ```
   create user INFACMI_User password 'Xxxx@xxx';
   ```

2. Create a new user role and grant it to the Data Ingestion and Replication user. Use the following SQL statements:

   ```
   create role INFACMI_superpipe;
   grant role INFACMI_superpipe to user INFACMI_User;
   ```

3. Grant usage on the Snowflake virtual warehouse to the new role. Use the following SQL statement:

   ```
   grant usage on warehouse warehouse_name to role INFACMI_superpipe;
   ```

4. Grant usage on the Snowflake database to the new role. Use the following SQL statement:

   ```
   grant usage on database INFACMI_DB1 to role INFACMI_superpipe;
   ```

5. Create a new schema. Use the following SQL statements:

   ```
   use database INFACMI_DB1;
   create schema sh_superpipe;
   ```

6. Grant create stream, create view, and create table privileges on the new Snowflake schema to the new role. Use the following SQL statement:

   ```
   grant create stream, create view, create table, usage on schema
   INFACMI_DB1.sh_superpipe to role INFACMI_superpipe;
   ```

7. Set the default role for the newly created user. Use the following SQL statement:

```
alter user INFACMI_User set default_role=INFACMI_superpipe;
```

8. Define a Snowflake Data Cloud connection to the target. You must use the **KeyPair** option as the authentication method. See Connectors and Connections > Snowflake Data Cloud connection properties.
9. Generate a private key with OpenSSL version 3.x.x and the PBE-SHA1-2DES or PBE-SHA1-3DES cipher. Use the following openssl commands to generate and format the private key:

```
openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -v1 PBE-SHA1-3DES -out
rsa_key.p8
```

10. Generate the public key. Use the following openssl command, in which the -in option references the file (rsa_key.p8) that contains the encrypted private key:

```
openssl rsa -in rsa_key.p8 -pubout -out rsa_key.pub
```

11. In Snowflake, assign the public key to the Snowflake user. Use the following SQL command:

```
alter user INFACMI_User set rsa_public_key='key_value’;
```

Next step: When you create an ingestion and replication task, select the **Superpipe** option on the **Target** page of the task wizard. You can also optionally specify a **Merge Frequency** value, which controls the frequency at which change data rows are merged and applied to the Snowflake target table.

Without Superpipe

If you do NOT use the Superpipe feature for Snowflake targets, complete the following steps as the ACCOUNTADMIN user:

1. Create a Data Ingestion and Replication user. Use one of the following SQL statements:

```
create user INFACMI_User password 'Xxxx@xxx';
```

or

```
replace user INFACMI_User password 'Xxxx@xxx';
```

2. Create a new role and grant the role to the Data Ingestion and Replication user. Use the following SQL statements:

```
create role INFA_CMI_Role;
grant role INFA_CMI_Role to user INFACMI_User;
```

3. Grant usage on the Snowflake virtual warehouse to the new role. Use the following SQL statement:

```
grant usage on warehouse CMIWH to role INFA_CMI_Role;
```

4. Grant usage on the Snowflake database to the new role. Use the following SQL statement:

```
grant usage, CREATE SCHEMA on database CMIDB to role INFA_CMI_Role;
```

5. Set the default role for the newly created user. Use the following SQL statement:

```
alter user INFACMI_User set default_role=INFA_CMI_Role;
```

Also, as the INFACMI_User, create a new schema:

```
create schema CMISchema;
```

Note: If the user's default role is used for ingestion and replication tasks and does not have the required privileges, the following error will be issued at runtime:

```
SQL compilation error: Object does not exist, or operation cannot be performed.
```

Usage considerations

- Database Ingestion and Replication provides alternative methods of moving data to a Snowflake Data Cloud target:
  - If you select the **Superpipe** option when defining an ingestion task, the ingestion job uses the Snowpipe Streaming API to stream rows of data directly to the target tables with low latency. This method is available for all load types. You must use **KeyPair** authentication.

- If you do not use Superpipe, ingestion jobs first write the data to data files in an internal stage, which has the name you specify in the task definition.

- If you do not use Superpipe and the internal stage specified in the target properties for an ingestion job does not exist, Database Ingestion and Replication automatically creates the stage by running the following SQL command:

  ```
  Create stage if not exists "Schema"."Stage_Bucket"”
  ```

  For the command to run successfully, the following privilege must be granted to your user role:

  ```
  GRANT CREATE STAGE ON SCHEMA "Schema" TO ROLE <your_role>;
  ```

- When you define a connection for a Snowflake Data Cloud target, you must set the **Additional JDBC URL Parameters** field to `database=target_database_name`. Otherwise, when you try to define the target in the database ingestion and replication task wizard, an error message reports that the list of schemas cannot be retrieved.

- When you define a connection for a Snowflake target using the **KeyPair** option as the authentication method and you generate the private key with OpenSSL 3.x.x version, use `PBE-SHA1-2DES` or `PBE-SHA1-3DES` cipher while generating the private key. Run one of the following commands:

  ```
  openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -v1 PBE-SHA1-3DES -out
  rsa_key.p8
  ```

  or:

  ```
  openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -v1 PBE-SHA1-2DES -out
  rsa_key.p8
  ```

  If you use a generic command without the `PBE-SHA1-2DES` or `PBE-SHA1-3DES` cipher, an error message about an invalid or unsupported private key might occur while fetching the target schema in the target definition step of the database ingestion and replication task wizard.

  The error message does not occur if you use OpenSSL 1.1.1 to generate the private key.

- Database ingestion and replication incremental load jobs and combined initial and incremental load jobs generate a recovery table named INFORMATICA_CDC_RECOVERY on the target to store internal service information that prevents jobs restarted after a failure from propagating previously processed data again. This recovery table is generated in the same schema as the target tables.

- For Snowflake targets, you cannot alter the scale of NUMBER fields or change the data type of an existing field to a different data type because Snowflake does not support these actions.

- If the source tables contain columns that have special characters in the name, you can replace the special characters with a provided choice of string in the Snowflake target.
  To replace the special characters with a provided choice of string in the generated target schema, perform the following actions before deploying the database ingestion and replication task:

  1. Create a properties file to be used at the metadata-manager layer. Add the `targetReplacementValue=<string>` key-value pair to the properties file.
  2. Open the **Runtime Environment** page in Administrator and edit the Secure Agent. Create a custom property in the **Custom Configuration Details** area:
     - Select the **Database Ingestion** service.
     - Select the **DBMI_AGENT_ENV** type.
     - Enter DBMI_TASK_OVERRIDE_PROPERTIES as the property name.
     - Enter the properties file location as the property value.
  3. Set the targetReplacementValue=<string> custom property in the **Target** page of the task wizard.

     Before running the task, add <jobname>. to the targetReplacementValue key in the properties file:

     ```
     <jobname>.targetReplacementValue=<string>
     ```

If the property affects all jobs, add "alljobs." to the targetReplacementValue key:

```
alljobs.targetReplacementValue=<string>
```

Configure private connectivity to Snowflake

You can access Snowflake using AWS or Azure Private Link endpoints.

The AWS or Azure Private Link setup ensures that the connection to Snowflake uses the AWS or Azure internal network and does not take place over the public Internet.

To connect to the Snowflake account over the private AWS network, see AWS Private Link and Snowflake.

To connect to the Snowflake account over the private Azure network, see Azure Private Link and Snowflake.

Default directory structure for CDC files on Amazon S3, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, and Microsoft Fabric OneLake targets

Database ingestion and replication jobs create directories on Amazon S3, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, and Microsoft Fabric OneLake targets to store information about change data processing.

The following directory structure is created by default on the targets:

```
Bucket
└──connection_folder
    └──job_folder
        ├──cycle
        │   ├──completed
        │   │   ├──completed_cycle_folder
        │   │   │   └──Cycle-timestamp.csv
        │   │   │       ...
        │   │   └──completed_cycle_folder
        │   │       └──Cycle-timestamp.csv
        │   └──contents
        │       ├──cycle_folder
        │       │   └──Cycle-contents-timestamp.csv
        │       │       ...
        │       └──cycle_folder
        │           └──Cycle-contents-timestamp.csv
        └──data
            └──table_name
                ├──data
                │   ├──cycle_folder
                │   │   └──table_name_timestamp.csv
                │   │       ...
                │   └──cycle_folder
                │       └──table_name_timestamp.csv
                └──schema
                    └──V1
                        └──table_name.schema
```

The following table describes the directories in the default structure:

Folder	Description
connection_folder	Contains the Database Ingestion and Replication objects. This folder is specified in the **Folder Path** field of the Amazon S3 connection properties, in the **Directory Path** field of the Microsoft Azure Data Lake Storage Gen2 connection properties, or in the **Lakehouse Path** field of the Microsoft Fabric OneLake connection properties. **Note:** This folder is not created for Google Cloud Storage targets.
job_folder	Contains job output files. This folder is specified in the **Directory** field on the **Target** page of the database ingestion and replication task wizard.
cycle/completed	Contains a subfolder for each completed CDC cycle. Each completed cycle subfolder contains a completed cycle file.
cycle/contents	Contains a subfolder for each CDC cycle. Each cycle subfolder contains a cycle contents file.
data	Contains output data files and schema files for each table.
data/*table_name*/schema/V1	Contains a schema file. **Note:** Database Ingestion and Replication does not save a schema file in this folder if the output files use the Parquet format.
data/*table_name*/data	Contains a subfolder for each CDC cycle that produces output data files.

Cycle directories

Database Ingestion and Replication uses the following pattern to name cycle directories:

```
[dt=]yyyy-mm-dd-hh-mm-ss
```

The "dt=" prefix is added to cycle folder names if you select the **Add Directory Tags** check box on the **Target** page of the database ingestion and replication task wizard.

Cycle contents files

Cycle contents files are located in cycle/contents/*cycle_folder* subdirectories. Cycle contents files contain a record for each table that has had a DML event during the cycle. If no DML operations occurred on a table in the cycle, the table does not appear in the cycle contents file.

Database Ingestion and Replication uses the following pattern to name cycle content files:

```
Cycle-contents-timestamp.csv
```

A cycle contents csv file contains the following information:

- Table name
- Cycle name
- Path to the cycle folder for the table
- Start sequence for the table
- End sequence for the table
- Number of Insert operations
- Number of Update operations
- Number of Delete operations
- *For combined load jobs only*: Number of Truncate operations

- *For combined load jobs only*: Number of Insert operations encountered during the initial load phase
- *For combined load jobs only*: Number of Delete operations encountered during the initial load phase
- Schema version
- Path to the schema file for the schema version

  Note: If the output data files use the Parquet format, Database Ingestion and Replication does not save a schema file at the path that is specified in the cycle contents file. Instead, you can use the schema file in the folder that is specified in the **Avro Schema Directory** field on the **Target** page of the database ingestion and replication task wizard.

Completed cycle files

Completed cycle files are located in cycle/completed/*completed_cycle_folder* subdirectories. A database ingestion and replication job creates a cycle file in this subdirectory after a cycle completes. If this file is not present, the cycle has not completed yet.

Database Ingestion and Replication uses the following pattern to name completed cycle files:

```
Cycle-timestamp.csv
```

A completed cycle csv file contains the following information:

- Cycle name
- Cycle start time
- Cycle end time
- Current sequence number at the time the cycle ended
- Path to the cycle contents file
- Reason for the end of cycle
  Valid reason values are:
  - NORMAL_COMMIT. A commit operation was encountered after the cycle had reached the DML limit or the end of the cycle interval. A cycle can end only on a commit boundary.
  - NORMAL_EXPIRY. The cycle ended because the cycle interval expired. The last operation was a commit.
  - *For combined initial load jobs only*: BACKLOG_COMPLETED. The cycle ended because CDC backlog processing completed. The CDC backlog consists of events captured during the initial load phase of the combined job. The backlog includes potential DML changes captured at the beginning or end of the initial load phase and during the transition from the initial load phase to the main CDC incremental processing.
  - *For combined load jobs only*: INITIAL_LOAD_COMPLETED. The cycle ended because the initial load completed.
  - *For combined load jobs only*: RESYNC_STARTED. The cycle ended because the table resync initiated.

Output data files

The data files contain records that include the following information:

- Operation type. Valid values are:
  - I for Insert operations.
  - U for Update operations.
  - D for Delete operations.
  - *For combined load jobs only*: T for Truncate operations.

- *For combined load jobs only*: X for Delete operations encountered during the initial load phase of a combined load job
- *For combined load jobs only*: Y for Insert operations encountered during the initial load phase of a combined load job

- Sortable sequence number. In combined initial and incremental load jobs, the sortable sequence number contains a 20-digit prefix that can be used to align rows with the resync version and the load job. The prefix is a combination of the following attributes:
  1. Incarnation. This nine-digit number is incremented each time the table is resynced. The initial value is 1.
  2. Schema version. This nine-digit number is incremented each time a schema drift change is propagated for the table. The initial value is 1.
  3. Phase. This two-digit number changes when transition from unload, to backlog, to CDC is performed. Valid values are:
     - 00 for Truncation, which is the first data record written during initial load or resync
     - 01 for a normal insert during initial load or resync
     - 02 for a change detected during the initial load
     - 03 for a change detected after the initial load or resync is completed but before the transition back to the main CDC phase
     - 04 for a change detected during the normal CDC phase
- Data columns

  Note: Insert and Delete records contain only after images. Update records contain both before and after images.

Custom directory structure for output files on Amazon S3, Google Cloud Storage, Flat File, Microsoft Fabric OneLake, and ADLS Gen2 targets

You can configure a custom directory structure for the output files that initial load, incremental load, and combined initial and incremental load jobs write to Amazon S3, Google Cloud Storage, Flat File, Microsoft Azure Data Lake Storage (ADLS) Gen2, or Microsoft Fabric OneLake targets if you do not want to use the default structure.

Initial loads

By default, initial load jobs write output files to *tablename_timestamp* subdirectories under the parent directory. For Amazon S3, Flat File, and ADLS Gen2 targets, the parent directory is specified in the target connection properties if the **Connection Directory as Parent** check box is selected on the **Target** page of the task wizard.

- In an Amazon S3 connection, this parent directory is specified in the **Folder Path** field.
- In a Flat File connection, the parent directory is specified in the **Directory** field.
- In an ADLS Gen2 connection, the parent directory is specified in the **Directory Path** field.

For Google Cloud Storage targets, the parent directory is the bucket container specified in the **Bucket** field on the **Target** page of the task wizard.

For Microsoft Fabric OneLake targets, the parent directory is the path specified in the **Lakehouse Path** field in the Microsoft Fabric OneLake connection properties.

You can customize the directory structure to suit your needs. For example, for initial loads, you can write the output files under a root directory or directory path that is different from the parent directory specified in the connection properties to better organize the files for your environment or to find them more easily. Or you can consolidate all output files for a table directly in a directory with the table name rather than write the files to separate timestamped subdirectories, for example, to facilitate automated processing of all of the files.

To configure a directory structure, you must use the **Data Directory** field on the **Target** page of the ingestion task wizard. The default value is `{TableName}_{Timestamp}`, which causes output files to be written to *tablename_timestamp* subdirectories under the parent directory. You can configure a custom directory path by creating a directory pattern that consists of any combination of case-insensitive placeholders and directory names. The placeholders are:

- {TableName} for a target table name
- {Timestamp} for the date and time, in the format yyyymmdd_hhmissms, at which the initial load job started to transfer data to the target
- {Schema} for the target schema name
- {YY} for a two-digit year
- {YYYY} for a four-digit year
- {MM} for a two-digit month value
- {DD} for a two-digit day in the month

A pattern can also include the following functions:

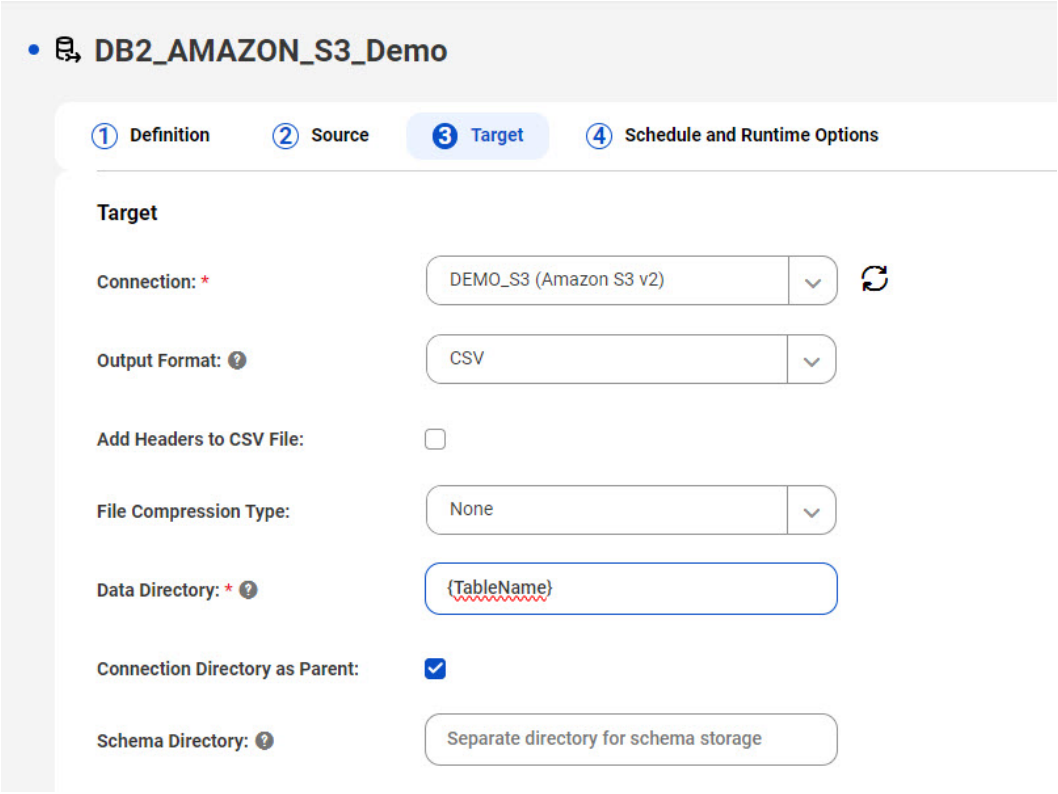
- toLower() to use lowercase for the values represented by the placeholder in parentheses
- toUpper() to use uppercase for the values represented by the placeholder in parentheses

By default, the target schema is also written to the data directory. If you want to use a different directory for the schema, you can define a directory pattern in the **Schema Directory** field.

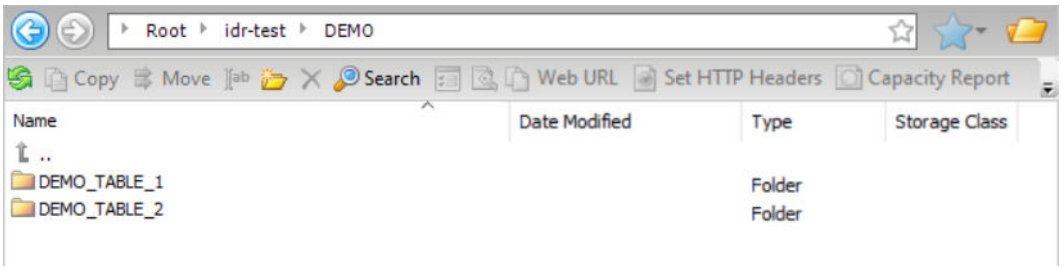
Example 1

You are using an Amazon S3 target and want to write output files and the target schema to the same directory, which is under the parent directory specified in the **Folder Path** field of the connection properties. In this case, the parent directory is `idr-test/DEMO/`. You want to write all of the output files for a table to a directory that has a name matching the table name, without a timestamp. You must complete the **Data**

Directory field and select the **Connection Directory as Parent** check box. The following image shows this configuration on the **Target** page of the task wizard:



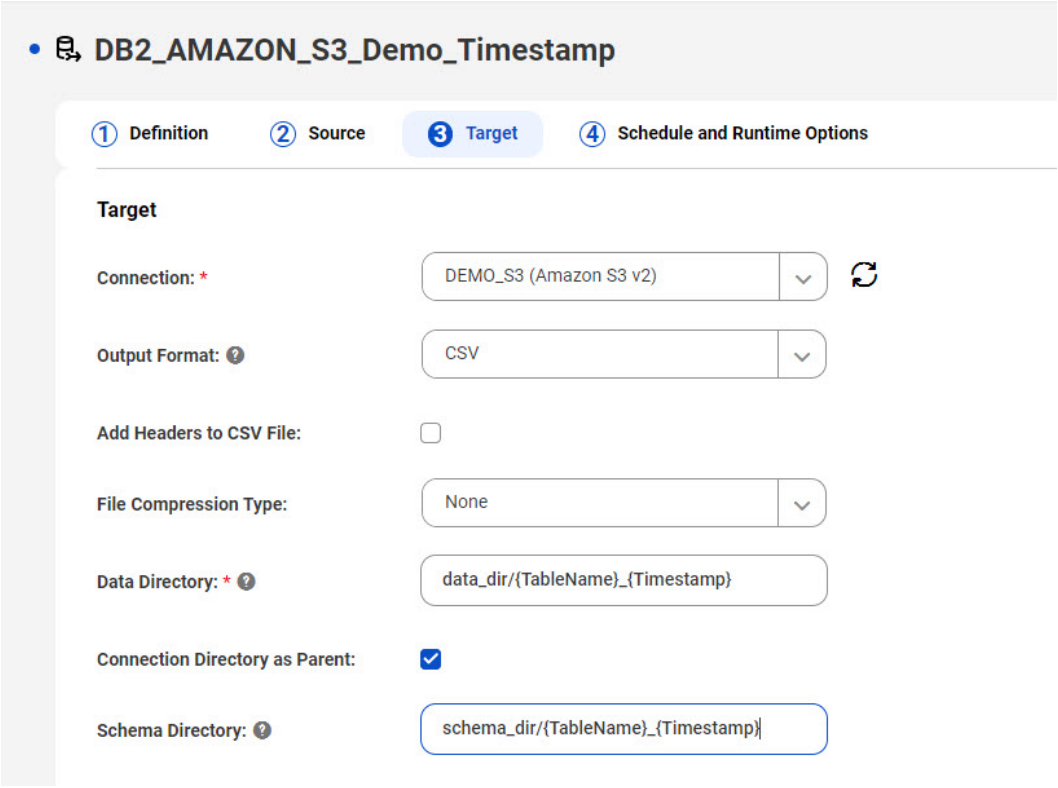
Based on this configuration, the resulting directory structure is:



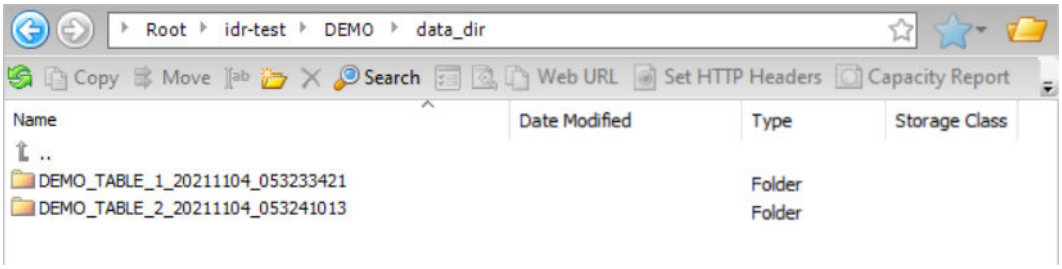
Example 2

You are using an Amazon S3 target and want to write output data files to a custom directory path and write the target schema to a separate directory path. To use the directory specified in the **Folder Path** field in the Amazon S3 connection properties as the parent directory for the data directory and schema directory, select **Connection Directory as Parent**. In this case, the parent directory is `idr-test/DEMO/`. In the **Data Directory** and **Schema Directory** fields, define directory patterns by using a specific directory name, such as data_dir and schema_dir, followed by the default {TableName}_{Timestamp} placeholder value. The placeholder

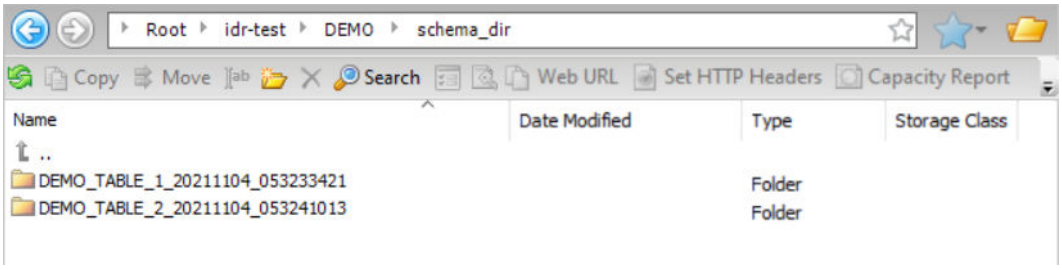
creates *tablename_timestamp* destination directories. The following image shows this configuration on the **Target** page of the task wizard:



Based on this configuration, the resulting data directory structure is:



And the resulting schema directory structure is:



Incremental loads and combined initial and incremental loads

By default, incremental load and combined initial and incremental load jobs write cycle files and data files to subdirectories under the parent directory. However, you can create a custom directory structure to organize the files to best suit your organization's requirements.

This feature applies to database ingestion incremental load and combined initial and incremental load tasks that have Amazon S3, Google Cloud Storage, Microsoft Fabric OneLake, or Microsoft Azure Data Lake Storage (ADLS) Gen2 targets.

For Amazon S3 and ADLS Gen2 targets, the parent directory is set in the target connection properties if the **Connection Directory as Parent** check box is selected on the **Target** page of the task wizard.

- In an Amazon S3 connection, the parent directory is specified in the **Folder Path** field.
- In an ADLS Gen2 connection, the parent directory is specified in the **Directory Path** field.

For Google Cloud Storage targets, the parent directory is the bucket container specified in the **Bucket** field on the **Target** page of the task wizard.

For Microsoft Fabric OneLake targets, the parent directory is the path specified in the **Lakehouse Path** field in the Microsoft Fabric OneLake connection properties.

You can customize the directory structure to suit your needs. For example, you can write the data and cycle files under a target directory for the task instead of under the parent directory specified in the connection properties. Alternatively, you can 1) consolidate table-specific data and schema files under a subdirectory that includes the table name, 2) partition the data files and summary contents and completed files by CDC cycle, or 3) create a completely customized directory structure by defining a pattern that includes literal values and placeholders. For example, if you want to run SQL-type expressions to process the data based on time, you can write all data files directly to timestamp subdirectories without partitioning them by CDC cycle.

To configure a custom directory structure for an incremental load task, define a pattern for any of the following optional fields on the **Target** page of the ingestion task wizard:

Field	Description	Default
Task Target Directory	Name of a root directory to use for storing output files for an incremental load task. If you select the **Connection Directory as Parent** option, you can still optionally specify a task target directory. It will be appended to the parent directory to form the root for the data, schema, cycle completion, and cycle contents directories. This field is required if the {TaskTargetDirectory} placeholder is specified in patterns for any of the following directory fields.	None
Connection Directory as Parent	Select this check box to use the parent directory specified in the connection properties. This field is not available for the Microsoft Fabric OneLake target.	Selected
Data Directory	Path to the subdirectory that contains the data files. In the directory path, the {TableName} placeholder is required if data and schema files are *not* partitioned by CDC cycle.	{TaskTargetDirectory}/data/{TableName}/data
Schema Directory	Path to the subdirectory in which to store the schema file if you do not want to store it in the data directory. In the directory path, the {TableName} placeholder is required if data and schema files are not partitioned by CDC cycle.	{TaskTargetDirectory}/data/{TableName}/schema
Cycle Completion Directory	Path to the directory that contains the cycle completed file.	{TaskTargetDirectory}/cycle/completed
Cycle Contents Directory	Path to the directory that contains the cycle contents files.	{TaskTargetDirectory}/cycle/contents

Field	Description	Default
Use Cycle Partitioning for Data Directory	Causes a timestamp subdirectory to be created for each CDC cycle, under each data directory. If this option is not selected, individual data files are written to the same directory without a timestamp, unless you define an alternative directory structure.	Selected
Use Cycle Partitioning for Summary Directories	Causes a timestamp subdirectory to be created for each CDC cycle, under the summary contents and completed subdirectories.	Selected
List Individual Files in Contents	Lists individual data files under the contents subdirectory. If **Use Cycle Partitioning for Summary Directories** is cleared, this option is selected by default. All of the individual files are listed in the contents subdirectory unless you can configure custom subdirectories by using the placeholders, such as for timestamp or date. If **Use Cycle Partitioning for Data Directory** is selected, you can still optionally select this check box to list individual files and group them by CDC cycle.	Not selected if **Use Cycle Partitioning for Summary Directories** is selected. Selected if you cleared **Use Cycle Partitioning for Summary Directories**.

A directory pattern consists of any combination of case-insensitive placeholders, shown in curly brackets { }, and specific directory names. The following placeholders are supported:

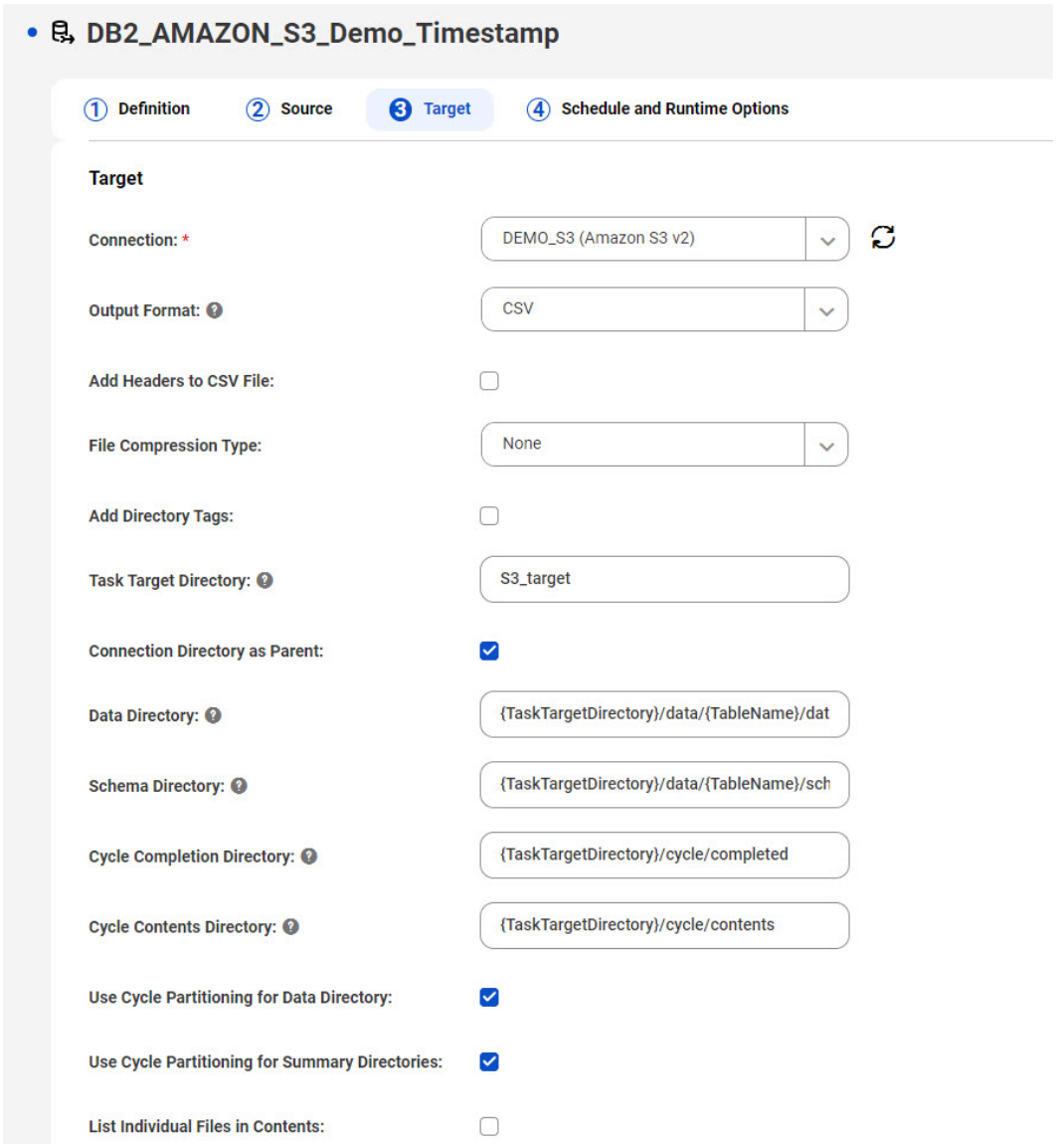
- {TaskTargetDirectory} for a task-specific base directory on the target to use instead of the directory the connection properties
- {TableName} for a target table name
- {Timestamp} for the date and time, in the format yyyymmdd_hhmissms
- {Schema} for the target schema name
- {YY} for a two-digit year
- {YYYY} for a four-digit year
- {MM} for a two-digit month value
- {DD} for a two-digit day in the month

Note: The timestamp, year, month, and day placeholders indicate when the CDC cycle started when specified in patterns for data, contents, and completed directories, or indicate when the CDC job started when specified in the schema directory pattern.

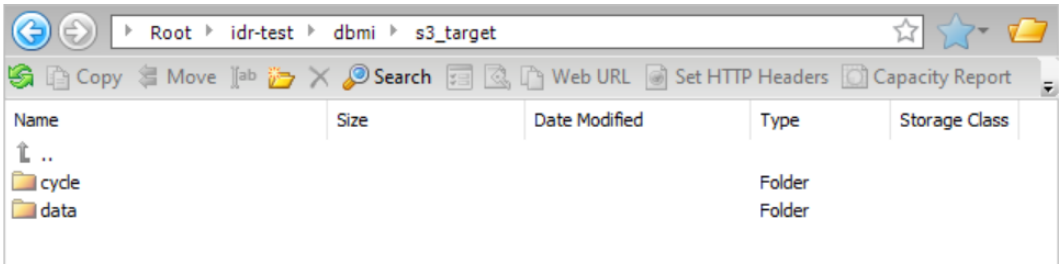
Example 1

You want to accept the default directory settings for incremental load or combined initial and incremental load jobs as displayed in the task wizard. The target type is Amazon S3. Because the **Connection Directory as Parent** check box is selected by default, the parent directory path that is specified in the **Folder Path** field of the Amazon S3 connection properties is used. This parent directory is `idr-test/dbmi/`. You also must specify a task target directory name, in this case, s3_target, because the {TaskTargetDirectory} placeholder is used in the default patterns in the subsequent directory fields. The files in the data directory and schema directory will be grouped by table name because the {TableName} placeholder is included in their default patterns. Also, because cycle partitioning is enabled, the files in the data directory, schema directory, and cycle summary directories will be subdivided by CDC cycle. The following image shows the default

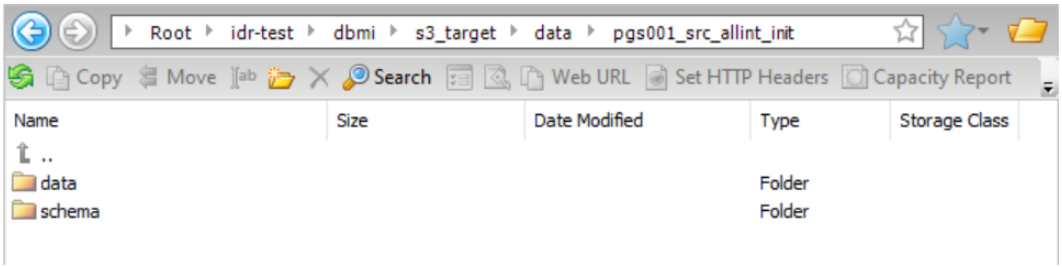
configuration settings on the **Target** page of the task wizard, except for the specified task target directory name:



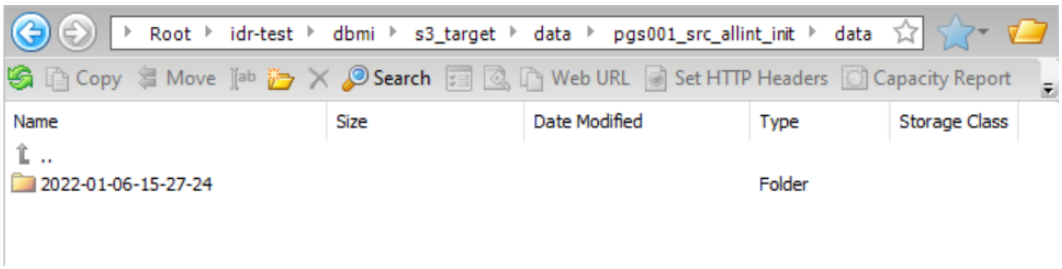
Based on this configuration, the resulting data directory structure is:



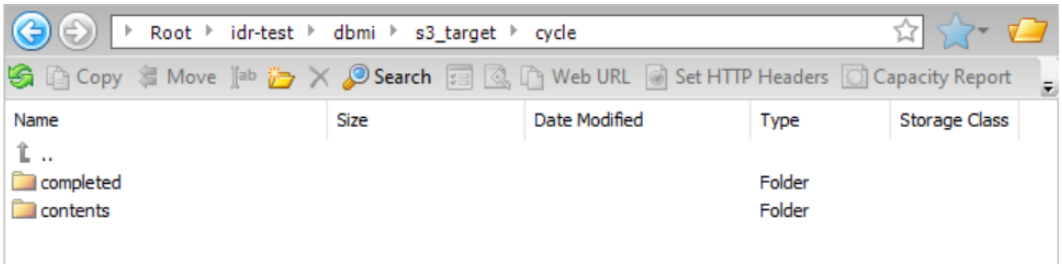
If you drill down on the data folder and then on a table in that folder (pgs001_src_allint_init), you can access the data and schema subdirectories:



If you drill down on the data folder, you can access the timestamp directories for the data files:



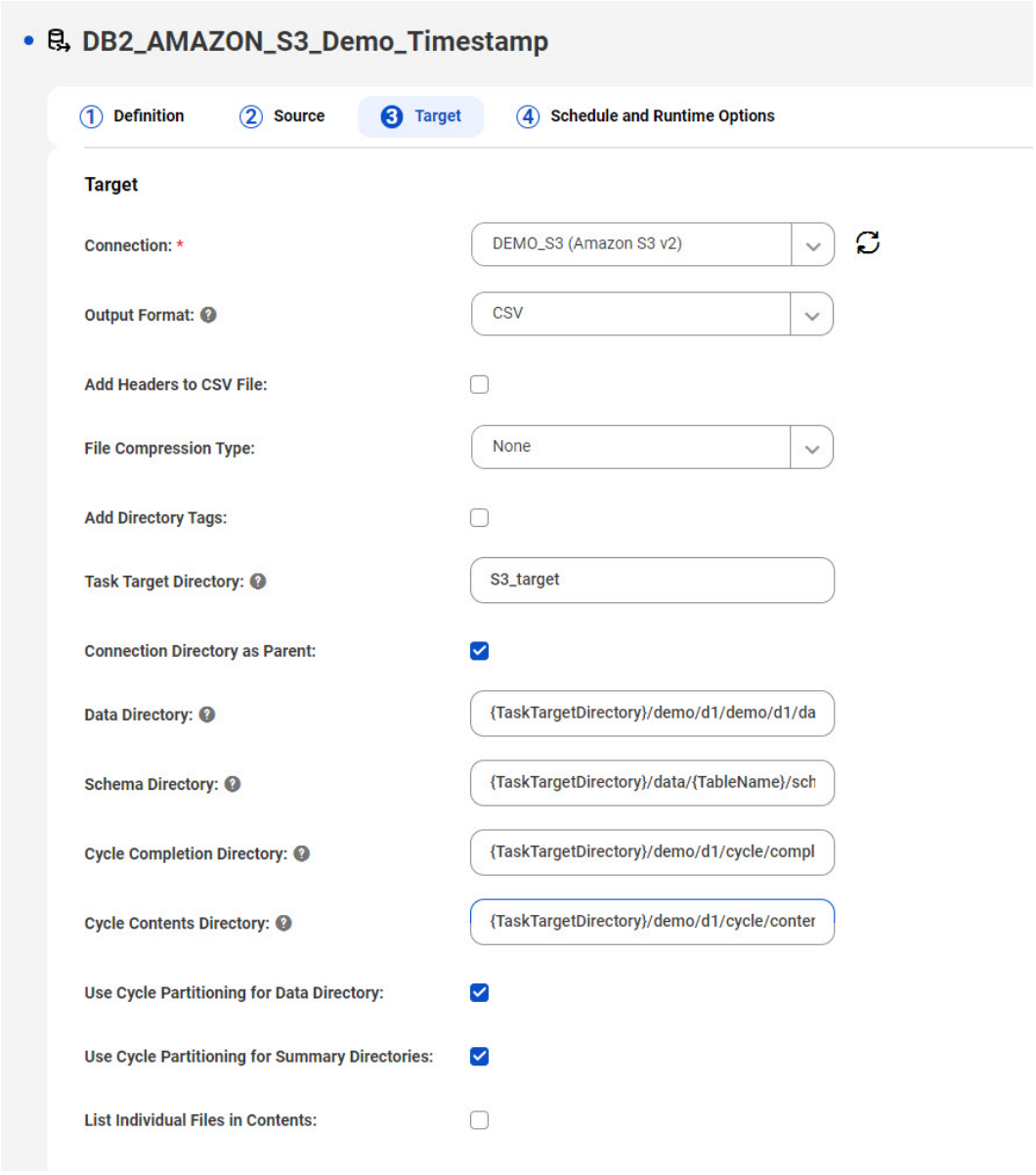
If you drill down on cycle, you can access the summary contents and completed subdirectories:



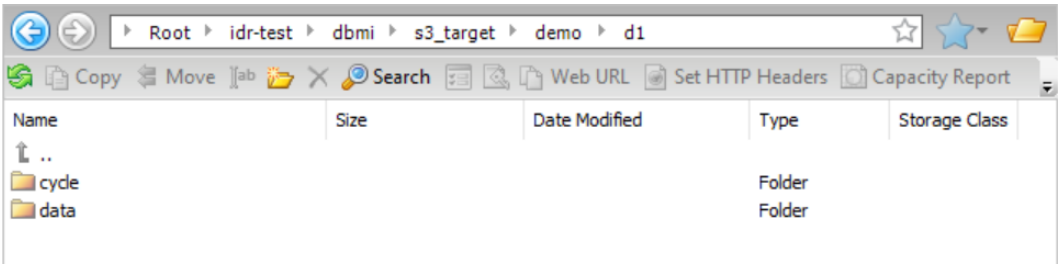
Example 2

You want to create a custom directory structure for incremental load or combined initial and incremental load jobs that adds the subdirectories "demo" and "d1" in all of the directory paths except in the schema directory so that you can easily find the files for your demos. Because the **Connection Directory as Parent** check box is selected, the parent directory path (`idr-test/dbmi/`) that is specified in the **Folder Path** field of the Amazon S3 connection properties is used. You also must specify the task target directory because the {TaskTargetDirectory} placeholder is used in the patterns in the subsequent directory fields. The files in the data directory and schema directory will be grouped by table name. Also, because cycle partitioning is

enabled, the files in the data, schema, and cycle summary directories will be subdivided by CDC cycle. The following image shows the custom configuration on the **Target** page of the task wizard:



Based on this configuration, the resulting data directory structure is:



Supported Avro data types

Database Ingestion and Replication supports some of the primitive and logical data types that Avro schemas provide. These data types pertain to target types that support Avro or Parquet output format.

A primitive data type represents a single data value. A logical data type is an Avro primitive or complex data type that has additional attributes to represent a derived type.

The following table lists the primitive Avro data types that Database Ingestion and Replication supports:

Primitive data type	Description
INT	32-bit signed integer
LONG	64-bit signed integer
FLOAT	Single precision (32-bit) IEEE 754 floating-point number
DOUBLE	Double precision (64-bit) IEEE 754 floating-point number
BYTES	Sequence of 8-bit unsigned bytes
STRING	Unicode character sequence

The following table lists the logical Avro data types that Database Ingestion and Replication supports:

Logical data type	Description
DECIMAL	An arbitrary-precision signed decimal number of the form unscaled × 10^{-scale}
DATE	A date, without reference to a time or time zone.
TIME	A time of day that has the precision of 1 millisecond or 1 microsecond, without reference to a time zone or date.
TIMESTAMP	A date and time value that has the precision of 1 millisecond or microsecond, without reference to a particular calendar or time zone.

For Databricks targets, Database Ingestion and Replication does not use the following data types in the intermediate Parquet files:

- TIMESTAMP, with millisecond precision
- TIME, with either millisecond or microsecond precision

Schema drift handling

Database Ingestion and Replication can be configured to automatically detect some source schema changes and handle these changes on the target. This process is referred to as *schema drift*.

Database Ingestion and Replication can detect the following types of source schema changes:

- Add column

- Modify column
- Drop column
- Rename column

When you define a task, on the **Schedule and Runtime Options** page of the database ingestion and replication task wizard, you can configure how the supported types of schema changes are handled. For example, you can configure schema drift options to ignore the changes, replicate them, or stop the job or subtask when a schema change occurs. For more information, see "Configuring schedule and runtime options" on page 159. Note that dIfferent types of schema changes might have different default settings, depending on the target type.

Schema drift options are supported for the following source - target combinations and load types:

Source	Load Type	Target
Db2 for i	Incremental Combined initial and incremental	Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka (incremental loads only), Microsoft Azure Data Lake Storage, Microsoft Azure Synapse Analytics, Oracle, Oracle Cloud Object Storage, PostgreSQL, Snowflake, and SQL Server
Db2 for LUW	Incremental Combined initial and incremental	Snowflake
Db2 for z/OS, except Db2 11	Incremental Combined initial and incremental	Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka (incremental loads only), Microsoft Azure Data Lake Storage, Microsoft Azure Synapse Analytics, Oracle, Oracle Cloud Object Storage, Snowflake, and SQL Server
Microsoft SQL Server	Incremental Combined initial and incremental	Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka (incremental loads only), Microsoft Azure Data Lake Storage, Microsoft Azure Synapse Analytics, Oracle, Oracle Cloud Object Storage, Snowflake, and SQL Server
Oracle	Incremental Combined initial and incremental	Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka (incremental loads only), Microsoft Azure Data Lake Storage, Microsoft Azure Synapse Analytics, Oracle, Oracle Cloud Object Storage, PostgreSQL, Snowflake, and SQL Server
PostgreSQL	Incremental Combined initial and incremental	Incremental loads: Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka (incremental loads only), Microsoft Azure Data Lake Storage, Microsoft Azure Synapse Analytics, Oracle, Oracle Cloud Object Storage, and Snowflake Combined initial and incremental loads: Oracle and Snowflake

Considerations:

- If you try to replicate a type of schema change that is not supported on the target, the database ingestion and replication job ends with an error.
- Database Ingestion and Replication does not replicate source changes that add, remove, or modify primary key or unique key constraints. If these types of changes occur on the source, you must resynchronize the target tables.
- If you configured schema drift options to stop the job when Database Ingestion and Replication detects a schema change, you can use the **Resume With Options** command to resume the job with an override schema drift option.

- Database Ingestion and Replication detects a schema change in a source table only after DML operations occur on the altered source table. If multiple schema changes occur without intervening DML operations, Database Ingestion and Replication detects all of the schema changes at one time, when a DML operation occurs. To ensure that Database Ingestion and Replication detects all of the supported schema changes correctly, Informatica recommends that you apply schema changes to source tables one by one, each followed by at least one DML change.
- If you set the **Add Column** option to **Replicate** for tasks that have an Oracle or SQL Server source, Database Ingestion and Replication assumes that any new column is added at the end of the table definition. If a new column appears in the middle of a table definition, Database Ingestion and Replication might treat the change as if the column has been dropped and added again. In this situation, the following alert might report that the same column has been both dropped and added:

  ```
  Column <column_name2> has been added. Column <column_name2> has been dropped.
  ```
- Database ingestion and replication tasks that have Microsoft Azure Synapse Analytics targets cannot replicate rename operations on source columns. The **Replicate** option is not available.
- Database ingestion and replication tasks that have Snowflake targets support modify operations on source columns with the following limitations:
  - Snowflake targets cannot modify the scale of NUMBER columns.
  - Snowflake targets do not support changing the data type of an existing column to a different data type.
- Database ingestion and replication tasks that have Google BigQuery targets cannot replicate rename or modify operations on source columns. The schema drift options for these operations are not available.
- If you set the **Add Column** option to **Replicate** and then add a column with a default value to a Db2 for i source table, Database Ingestion and Replication adds the default value to the newly added table rows to the target. However, existing rows on the target are not updated to reflect the default values. To get the default value populated to the existing target rows, perform another initial load to re-materialize the target.

Apply an audit history of all source table DML changes to target tables

You can configure database ingestion and replication incremental load and combined initial and incremental load tasks that have Databricks, Google BigQuery, Oracle, or Snowflake targets to write an audit trail of every DML change operation made on the source tables to the target. A row for each DML change on a source table is written to the generated target table along with the audit columns you select. The audit columns contain metadata about the change, such as the DML operation type, time, owner, transaction ID, generated ascending sequence number, and before image.

When you define the task, select **Audit** in the **Apply Mode** field on the **Target** page. The **Apply Mode** field is available for new or undeployed tasks.

To specify the audit metadata columns to add, select one or more of the check boxes under **Advanced** on the **Target** page:

- **Add Last Replicated Time**. Adds a column that records the timestamp at which a record was inserted or last updated in the target table. For initial loads, all loaded records have the same timestamp, except for Snowflake targets that use the Superpipe option where the seconds or minutes might vary slightly. For incremental loads and combined initial and incremental loads, the column records the timestamp of the last DML operation that was applied to the target.

  Note: This option is available for Google BigQuery and Snowflake targets only.

- **Add Operation <*metadata_type*>**. Adds columns that contain metadata for change operations, such as the DML operation type, time, owner, transaction ID, and generated ascending sequence number. The columns are populated when data is loaded to the target tables.
- **Add Before Images**. Adds _OLD columns that contain before-image data for Updates. You can compare the old and new column values in the tables.
- **Prefix for Metadata Columns**. Add a prefix to the names of the added audit columns to differentiate them from other table columns. Default is INFA_.

These fields are optional. Only the **Add Operation Type** check box is selected by default to add a column that shows the DML operation type: I (insert), D (delete), or U (update).

The first time you run a job associated with the task, the job generates the target tables with the selected audit metadata columns. Ensure that no constraints other than indexes exist on the target tables.

For targets that support **Soft Deletes** mode, including Databricks and Snowflake targets, for each delete operation processed as a soft delete, the operation type of "D" appears in the INFA_OPERATION_TYPE column and values are written to any other metadata columns that you selected. However, for update and insert operations, the INFA_OPERATION_TYPE column and all other selected metadata columns are NULL.

Example for Audit mode

For example, assume the following DML change operations occur on a source table in the order shown:

```
Insert into tableA pkey = 1
Update tableA where pkey=1
Update tableA where pkey=1
Delete from tableA where pkey = 1
```

All of the following rows appear in the target table, providing an audit trail of all of the source DML changes:

```
opType=I, pkey=1….
opType=U, pkey=1...
opType=U, pkey=1...
opType=D, pkey=1...
```

In this example, the only audit column selected is opType.

Note that when the task's apply mode is Standard, none of these rows appears on the target table because the last DML operation is a Delete, which supersedes the prior changes.

Note: If a combined initial and incremental load job captures an incremental insert change record during the initial unload phase, the job manufactures a delete for the same row to remove any duplicate that might have been obtained from the initial unload. This manufactured activity will be reflected in audit apply mode.

Ability to apply deletes as soft deletes on the target

For database ingestion and replication incremental load and combined initial and incremental load jobs that have any supported source type and a Databricks or Snowflake target, you can configure the task to process delete operations on the source as soft deletes on the target.

A soft delete marks a deleted row as deleted without actually removing it from the database. The row is applied to the target with the value of "D" in the generated INFA_OPERATION_TYPE metadata column.

Important: You can use Soft Deletes only if all of the source objects have primary keys and the source does not allow the primary key values to change in any row after the row is first created. If the primary key values change, duplicate rows might be written to the target, causing the target to become corrupted.

Example scenario: Your organization wants to use soft deletes in a data warehouse to mark the rows that were deleted at the source while still retaining the rows for audit purposes.

To enable soft deletes, set the **Apply Mode** field to **Soft Deletes** on **Target** page of the task wizard when you configure the database ingestion and replication task.

Note: If a combined initial and incremental load job captures an incremental insert change record during the initial unload phase, the job manufactures a delete for the same row to remove any duplicate that might have been obtained from the initial unload. This manufactured activity will be reflected in the soft delete apply mode.

Configuring a database ingestion and replication task

In Data Integration, use the database ingestion and replication task wizard to configure a database ingestion and replication task.

On the wizard pages, complete the following configuration tasks:

1. Define basic task information, such as the task name, project location, runtime environment, and load type.
2. Configure the source.
3. Configure the target.
4. Configure runtime options.

Click **Next** or **Back** to navigate from one page to another. At any point, you can click **Save** to save the information that you have entered so far.

After you complete all wizard pages, save the information and then click **Deploy** to make the task available as an executable job to the Secure Agent.

Before you begin

Before you begin, complete the following prerequisite tasks in Administrator:

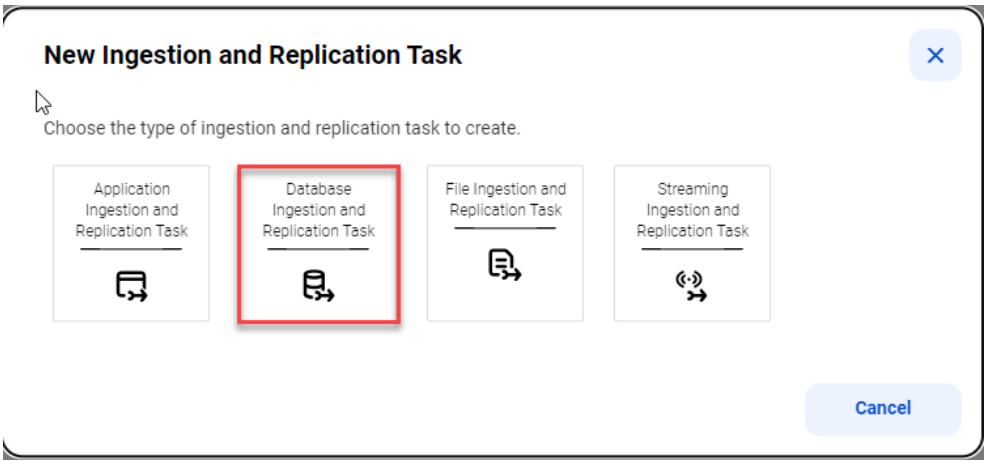
- Check that your organization has licenses for Database Ingestion and Replication and the DBMI packages.
- Verify that the Secure Agent in your runtime environment is running and that you can access the Data Ingestion and Replication service.
- Define the source and target connections.

Also, if you plan to perform incremental load operations with Oracle sources, ensure that the ORACLE_HOME environment variable is defined on the Secure Agent system.

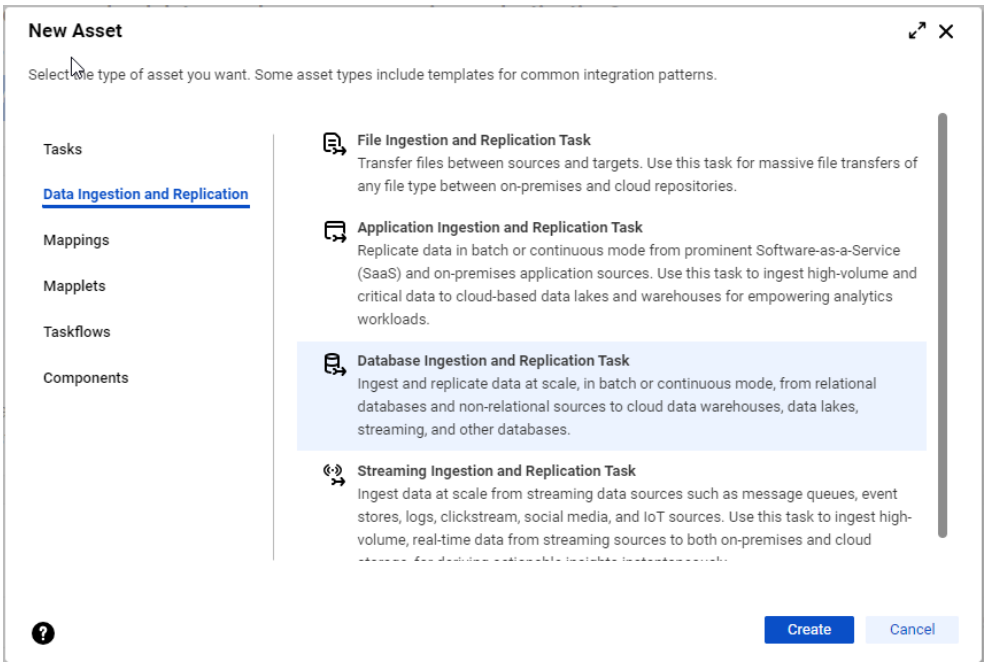
Defining basic task information

To begin defining a database ingestion and replication task, you must first enter some basic information about the task, such as a task name, project or project folder location, and load operation type.

1. Start the task wizard in one of the following ways:
   - On the Home page, click the **Ingest** panel and select **Database Ingestion and Replication Task**.



   - In the navigation bar on the **Explore** page or the Home page, click **New** to open the **New Asset** dialog box. Then, select **Data Ingestion and Replication** > **Database Ingestion and ReplicationTask** and click **Create**.



   Note: If your organization does not have a custom license for application, database, file, or streaming ingestion and replication, the **Data Ingestion and Replication** category still appears but you cannot select and configure an ingestion and replication task of the unlicensed task type or types.

The **Definition** page of the database ingestion and replication task wizard appears.

2. Configure the following properties:

Property	Description
Name	A name for the database ingestion and replication task. Task names can contain Latin alphanumeric characters, spaces, periods (.), commas (,), underscores (_), plus signs (+), and hyphens (-). Task names cannot include other special characters. Task names are not case sensitive. Maximum length is 50 characters. **Note:** If you include spaces in the database ingestion and replication task name, after you deploy the task, the spaces do not appear in the corresponding job name.
Location	The project or project\folder that will contain the task definition. The default is the currently selected project or project subfolder in Explore. If a project or project subfolder is not selected, the default is the **Default** project.
Runtime Environment	The runtime environment in which you want to run the task. The runtime environment must be a Secure Agent group that consists of one or more Secure Agents. A Secure Agent is a lightweight program that runs tasks and enables secure communication. For database ingestion and replication tasks, the Cloud Hosted Agent is not supported and does not appear in the **Runtime Environment** list. Serverless runtime environments are also not supported. **Tip:** Click the **Refresh** icon to refresh the list of runtime environments.
Description	An optional description for the task. Maximum length is 4,000 characters.
Load Type	The type of load operation that the database ingestion and replication task performs. Options are: - **Initial Load**. Loads data read at a specific point in time from source tables to a target in a batch operation. You can perform an initial load to materialize a target to which incremental change data will be sent. - **Incremental Load**. Propagates source data changes to a target continuously or until the job is stopped or ends. The job replicates the changes that have occurred since the last time the job ran or from a specific start point for the first job run. - **Initial and Incremental Loads**. Performs an initial load of point-in-time data to the target and then automatically switches to replicating incremental data changes made to the same source tables on a continuous basis. **Note:** If a change record is captured during the initial unload load phase, it's withheld from apply processing until after the unload phase completes. Any insert rows captured during the unload phase are converted into a pair of delete and insert operations so that only one insert row is applied to the target in the case where the insert occurs in both the unloaded data and the captured change data.

3. Click **Next**.

Configuring the source

Configure the source on the **Source** page of the database ingestion and replication task wizard.

Note: For MongoDB sources only, the task wizard displays *database* instead of *schema* and displays *collection* instead of *table*. However, for simplicity, the terms *schema* and *table* are used in this documentation to cover all source types.

1. In the **Connection** list, select the connection for the source system. The connection type appears in parentheses after the connection name.

   The connection must be predefined in Administrator for a runtime environment that your organization uses.

   The list includes only the connection types that are valid for the load type selected on the **Definition** page. No connections are listed if you did not select a load type.

   If you change the load type and the selected connection is no longer valid, a warning message is issued and the **Connection** field is cleared. You must select another connection that is valid for the updated load type.

   Note: After you deploy the database ingestion and replication task, you cannot change the connection without first undeploying the associated job. After you change the connection, you must deploy the task again.

2. In the **Schema** list, select the source schema that includes the source tables. If you specified a schema in the connection properties, it is selected by default but you can change it.

   The list includes only the schemas that are available in the database accessed with the specified source connection.
   When creating a task that has an Oracle, Microsoft SQL Server, Netezza, or PostgreSQL source, the schema name that is specified in the connection properties is displayed by default.

3. If you are defining a Db2 for i source for an incremental load task, in the **Journal Name** field, select the name of the journal that records the changes made to the source tables.

4. If you are defining a PostgreSQL source for an incremental load or combined initial and incremental load task, complete the following fields:

Field	Description
Replication Slot Name	The unique name of a PostgreSQL replication slot. A slot name can contain Latin alphanumeric characters in lowercase and the underscore (_) character. Maximum length is 63 characters. **Important:** Each database ingestion and replication task must use a different replication slot.
Replication Plugin	The PostgreSQL replication plug-in. Options are: - **pgoutput**. You can select this option only for PostgreSQL version 10 and later. - **wal2json**
Publication	If you selected **pgoutput** as the replication plug-in, specify the publication name that this plug-in uses. **Note:** This field is not displayed if you selected wal2json as the replication plug-in.

5. If you are defining an incremental load or combined initial and incremental load task that has a Db2 for LUW, Oracle, or SQL Server source, select the capture method that you want to use under **Change Data Capture Method**.

   a. In the **CDC Method** field, select one of the following options to indicate the method to use for capturing source changes:

Method	Supported Sources	Description
CDC Tables	SQL Server only	Read data changes directly from the SQL Server CDC tables. For SQL Server sources, this method provides the best replication performance and highest reliability of results.
Log-based	Oracle and SQL Server	Capture Inserts, Updates, Deletes, and column DDL changes in near real time by reading the database transaction logs. For Oracle sources, data changes are read from the Oracle redo logs. For SQL Server sources, data changes are read from the SQL Server transaction log and the enabled SQL Server CDC tables. Exception: For Azure SQL Database sources, data changes are read from CDC tables only.
Query-based	Db2 for LUW, Oracle, and SQL Server	Capture Inserts and Updates by using a SQL WHERE clause that points to a CDC query column. The query column is used to identify the rows that contain the changes made to the source tables since the beginning of the CDC interval. For Db2 for LUW sources in incremental load and initial and incremental load jobs, this capture method is the only available option.

   b. If you selected the **Query-based** option, complete the following additional fields:

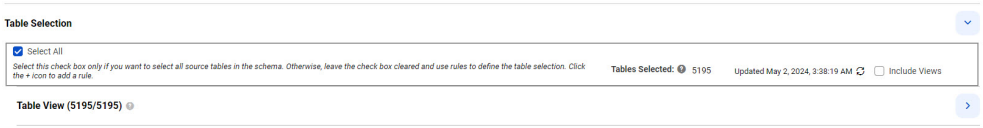
   - **CDC Query Column Type**. The column type of the CDC query column in the source table. The only available option is **Timestamp**.

     Note: "Timestamp" represents a column data type that combines the date and time. For Oracle, the supported data type for the query column is TIMESTAMP. For SQL Server, the supported data types for the query column are DATETIME and DATETIME2.

   - **CDC Query Column Name**. The case-sensitive name of the CDC query column in the source table. The column must be present in the source table. Maximum length is 70 characters.
   - **CDC Interval**. The frequency of a query-based change data capture cycle, expressed in days, hours, and minutes. You must enter a positive number in at least one of the interval fields. Otherwise, an error will be issued when you try to save the task. The default value is 5 minutes.

6. Under **Table Selection**, use one of the following methods to select source tables:

   - Select **Select All** to select all tables and columns in the schema for data replication. The **Tables Selected** field shows the count of all selected tables. You won't be able to edit the selection later under **Table View**.



   Note: Fetching information for all of the tables might take a long time.

- Under **Rules**, define rules to specify a subset of source tables to replicate. If you use rules, you'll be able to individually select or deselect tables and the columns in selected tables under **Table View**. You'll also be able to set an option for trimming spaces in character data. To add rules, go to 8.

7. If you want to include database views as sources, select the **Include Views** check box to the right of the Refresh icon. This check box is available only for initial load tasks that have a Db2 for i, Db2 for LUW, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, or Teradata source.

   The views will be fetched and included in the **Tables Selected** count and in the list of table names.

8. To add a table-selection rule, first make sure that the **Select All** check box is cleared. Then perform the following substeps:

   a. Under **Rules**, click the Add Rule (+) icon above the first table. A row is added to the table.

   b. In the **Table Rule** column, select **Include** or **Exclude** to create an inclusion or exclusion rule, respectively.

   c. In the **Condition** column, enter a table name or a table-name mask that includes one or more wildcards to identify the source tables to include in or exclude from table selection. Use the following guidelines:

      - A mask can contain one or both of the following wildcards: an asterisk (*) wildcard to represent one or more characters and a question mark (?) wildcard to represent a single character. A wildcard can occur multiple times in a mask value and can occur anywhere in the value.
      - The task wizard is case sensitive. Enter the table names or masks in the case with which the tables were defined.
      - Do not include delimiters such as quotation marks or brackets, even if the source database uses them. For example, Oracle requires quotation marks around lowercase and mixed-case names to force the names to be stored in lowercase or mixed case. However, in the task wizard, you must enter the lowercase or mixed-case names without quotation marks.
      - If a table name includes special characters such as a backslash (\), asterisk(*), dollar sign ($), caret (^), or question mark (?), escape each special character with a backslash (\) when you enter the rule.

   d. Define additional rules as needed.

      If you define multiple Include and Exclude rules, they'll be processed in the order in which they're listed, from top to bottom. Use the arrow icons to change the order. For an example of using multiple rules, see "Example of rules for selecting source tables" on page 117.

   e. When finished, click **Apply Rules**.

      The **Total Tables Selected** and **Table View** counts are updated. If you click the Refresh icon, the **Tables Affected** count for each rule is shown.

      The following image shows multiple rules defined on the **Source** page:

      After you apply rules, if you add, delete, or change rules, you must click **Apply Rules** again. Click the Refresh icon to update the table counts. If you delete all rules without clicking **Apply Rules**, a validation error will occur at deployment, even if the **Table View** list still lists tables. If you switch to **Select All**, the rules are no longer in effect and disappear.

9. To perform trim actions on character columns in the source tables selected based on rules, create column action rules in the second "Action" table under **Rules**.

   Note: You cannot create column action rules for MongoDB sources.

   a. Click the Add Rule (+) icon above the second table.

   b. In the **Action** column, select one of the following options:

      - **LTRIM**. Trims spaces to the left of character column values.
      - **RTRIM**. Trims spaces to the right of character column values.
      - **TRIM**. Trims spaces to the left of and to the right of character column values.

   c. In the **Condition** column, enter a column name or a column name mask that includes one or more asterisk (*) or question mark (?) wildcards. The value is matched against columns in the selected source tables to identify the columns to which the action applies.

   Note: You can define multiple rules for different action types or for the same action type with different conditions. The rules are processed in the order in which they're listed, from top to bottom. Use the arrow icons to change the order.

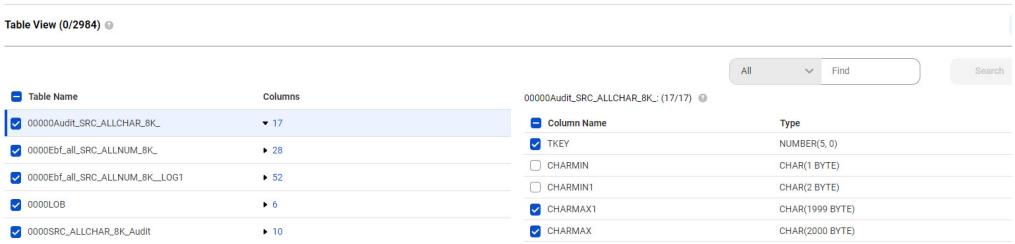
10. Under **Table View**, view or edit the set of selected source tables and columns.

    If you selected **Select All**, the lists of tables and columns are view only.

    If you applied rules, you can refine the set of selected tables by clicking the check box next to individual tables. Deselect any tables that you do not want to replicate, or select additional items to replicate. Click the Refresh icon to update the selected tables count.

    For Oracle and SQL Server sources only, you can also individually deselect or reselect the columns in a selected source table. To view or change the columns from which data will be replicated for a selected table, click the highlighted number of columns in the **Columns** column. The column names and data types are displayed to the right. By default, all columns are selected for a selected source table. To deselect a column or reselect it, click the check box next to the column name. You cannot deselect a primary key column.

    The following image shows selected tables and the selected columns for the first table:



Notes:

- To search for tables or columns or both, in the drop-down list above the columns list, select **Table Name**, **Columns**, or **All** and then enter a search string in the *Find* box and click **Search**. You can include a single asterisk (*) wildcard at the beginning or end of the string.
- The first time you change a check box setting for a table or column, the rules are no longer in effect. The selections under **Table View** take precedence. However, if you click the Add Rule (+) icon again, any tables that you deselected or selected individually are reflected as new rules in the Rules list and the rules once again take precedence. If you want to return to the **Table View** list, click **Apply Rules** again.
- In the **Table View** section, the length of character columns is shown in bytes. Since the number of bytes per character differs depending on the character set encoding used by the database, for some

sources, the actual column length in number of characters might be different than the number of bytes shown in the **Table View**.

- If you select columns individually, the resulting set of columns is fixed and will not be updated by any schema change, regardless of schema drift settings. For example, if a source column is added or renamed, that column is silently excluded from CDC processing because it's not in the list of selected columns. However, if a selected column is dropped on the source, the schema drift Drop Column option controls how it's handled. The dropped column operation is not reflected in the list of columns until you apply rules again.
- For combined initial and incremental load jobs, if you add to or modify a column selection, a resync operation is triggered automatically. The resync is required to ensure that the target table has the same values as the source. For incremental load jobs, which do not have a resync option, modifying a column selection results in an error. If you modify the column selection of a deployed job and then redeploy it, the source and target will no longer match.

  Note: A resync operation is not triggered, and no error is reported, if you remove a selected column. Removal of a selected column is treated the same as a drop column event, triggering your drop column schema drift settings for the task.

11. If you are defining an incremental load or combined initial and incremental load task that has a Db2 for i, Db2 for z/OS, Microsoft SQL Server, Oracle, PostgreSQL, SAP HANA, or SAP HANA Cloud source and one or more of the selected source tables are not enabled for change data capture, you can generate a script for enabling CDC and then run or download the script.

    Note: If you selected **Query-based** as the CDC method for Db2 for LUW, Oracle, or SQL Server sources, the **CDC Script** field is not available because it is not applicable for the query-based change capture method.

    a. In the **CDC Script** field, select one of the following options:

       - **Enable CDC for all columns**. Enables CDC for all columns in the selected source tables. This option is the only valid option for a Db2 for i, Db2 for z/OS, PostgreSQL, SAP HANA, SAP HANA Cloud, or SQL Server source.

         Note: For source tables without a primary key, including any tables with unique indexes, CDC is enabled for all columns by default, regardless of which option is selected.
       - **Enable CDC for primary key columns**. Enables CDC only for primary key columns in the selected source tables. Do not use this option for a Db2 for i, Db2 for z/OS, PostgreSQL, or SQL Server source or for any task that has a Google Big Query target.

       The script enables CDC in the following ways, depending on the source type:

       - For Db2 for i sources, the script enables journaling on the source tables.
       - For Db2 for z/OS sources, the script sets DATA CAPTURE CHANGES for source tables and certain Db2 catalog tables needed for CDC. After DATA CAPTURE CHANGES is set for one job, all other jobs recognize that the attribute is enabled in Db2 for the catalog tables needed because the Db2 catalog tables are a set of tables shared by all users of Db2.
       - For Microsoft SQL Server sources, the script runs the sys.sp_cdc_enable_db and sys.sp_cdc_enable_table stored procedures to enable CDC on the source database and tables. For RDS for SQL Server, the script runs the msdb.dbo.rds_cdc_enable_db procedure to enable CDC on the source database, and the sys.sp_cdc_enable_table script to track CDC for tables.
       - For Oracle sources, the script enables supplemental logging for all or primary key columns in the selected source tables to log additional information in the redo logs.

- For PostgreSQL sources, the script sets REPLICATION IDENTITY FULL on the selected source tables to write all column values to the WAL file. The script also creates a replication slot of the type of pgoutput or wal2json. If the slot type is pgoutput, the script also creates the publication and adds tables to it.
- For SAP HANA and SAP HANA Cloud sources, the script creates the required PKLOG, PROCESSED, and _CDC shadow tables. The script also creates three triggers and a sequence for each selected source table.

b. To run the script, click **Execute**.

If you do not have a database role or privilege that allows you to run the script, click the Download icon to download the script. The script file name has the following format: `cdc_script_taskname_number.txt`. Then ask your database administrator to run the script.

Make sure the script runs before you run the database ingestion and replication task.

Note: If you change to the **CDC Script** option later and run the script again, the script first drops CDC for the original set of columns and then enables CDC for the current set of columns. For SAP HANA sources, if the PROCESSED and PKLOG tables already exist, they are omitted from the new script. If the shadow _CDC table and triggers already exist for any selected table, the SQL statements for creating those objects are commented out in the new script.

12. For Microsoft SQL Server sources, complete the following fields:

- In the **Capture Filegroup** field, enter the name of the filegroup to be used for the change table that is created for the capture. If you leave this field empty, the change table is located in the default filegroup of the database.
- In the **Gating Role** field, enter the name of the database role that is used to gate access to change data. If you leave this field empty, the database does not use the gating role.

13. To create and download a list of the source tables that match the table selection criteria, perform the following substeps:

a. If you used rule-based table selection, in the **List Tables by Rule Type** list, select the type of selection rules that you want to use. Options are:

- **Include Rules Only**
- **Exclude Rules Only**
- **Include And Exclude Rules**

b. To include columns in the list, regardless of which table selection method you used, select the **Include Columns** check box.

Note: This option is not available for MongoDB sources.

c. Click the Download icon.

A downloaded list that includes columns has the following format:

```
status,schema_name,table_name,object_type,column_name,comment
```

The following table describes the information that is displayed in the downloaded list:

Field	Description
status	Indicates whether Database Ingestion and Replication excludes the source table or column from processing because it has an unsupported type. Valid values are: - **E**. The object is excluded from processing by an Exclude rule. - **I**. The object is included in processing. - **X**. The object is excluded from processing because it is an unsupported type of object. For example, unsupported types of objects include columns with unsupported data types and tables that include only unsupported columns. The comment field provides detail on unsupported types.
schema_name	Specifies the name of the source schema.
table_name	Specifies the name of the source table.
object_type	Specifies the type of the source object. Valid values are: - **C**. Column. - **T**. Table.
column_name	Specifies the name of the source column. This information appears only if you selected the **Columns** check box.
comment	Specifies the reason why a source object of an unsupported type is excluded from processing even though it matches the selection rules.

14. Under **Advanced**, set the advanced properties that are available for your source type and load type.

Property	Source and Load Type	Description
Disable Flashback	Oracle sources - Initial loads	Select this check box to disable Database Ingestion and Replication use of Oracle Flashback when fetching data from the database. The use of Oracle Flashback requires users to be granted the EXECUTE ON DBMS_FLASHBACK privilege, which is not necessary for initial loads. This check box is selected by default for new initial load tasks. For existing initial load tasks, this check box is cleared by default, which causes Oracle Flashback to remain enabled. For tasks that have partitioning enabled, this check box is automatically selected and unavailable for editing.

Include LOBs	Oracle sources:
	- Initial loads, incremental loads, and combined initial and incremental loads to Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen 2, Microsoft Azure Synapse Analytics, Microsoft Fabric OneLake, Oracle, Oracle Cloud Object Storage, PostgreSQL, Snowflake, or SQL Server targets.
	Incremental loads and combined loads can use either the **Log-based** or **Query-based** CDC method. However, jobs that use the**Log-based** CDC method do not replicate data from LONG, LONG RAW, and XML columns to the generated target columns.
	Db2 for LUW sources:
	- Initial loads, incremental loads, and combined loads to Microsoft Azure Data Lake Storage Gen 2, Microsoft Azure Synapse Analytics, or Snowflake targets. Incremental loads and combined loads must use the **Query-based** CDC method.
	PostgreSQL sources:
	- Initial loads to Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen 2, Microsoft Azure Synapse Analytics, Microsoft Fabric OneLake, Oracle, Oracle Cloud Object Storage,
		Select this check box if the source contains the large-object (LOB) columns from which you want to replicate data to a target.
		LOB data types:
		- For Db2 for LUW: BLOB, CLOB, DBCLOB, LONG VARCHAR, LONG VARCHAR FOR BIT, LONG VARGRAPHIC, and XML
		- For Oracle: BLOB, CLOB, NCLOB, LONG, LONG RAW, and XML
		- For PostgreSQL: BYTEA, TEXT, and XML plus some other potentially large types such as JSON, JSONB
		- For SQL Server: GEOGRAPHY, GEOMETRY, IMAGE, NTEXT, NVARCHAR(MAX), TEXT, VARBINARY(MAX), VARCHAR(MAX), and XML
		LOB data might be truncated, primarily depending on the maximum size that the target allows.
		Target-side truncation points:
		- BLOB, BYTEA, GEOGRAPHY, GEOMETRY, IMAGE, LONG RAW, LONG VARCHAR FOR BIT, or VARBINARY(MAX) columns are truncated before being written to BINARY columns on the target.
		  - For Amazon S3, Databricks, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, Oracle, Oracle Cloud Object Storage, Microsoft Fabric OneLake, PostgreSQL, and SQL Server targets, the data is truncated to 16777216 bytes.
		  - For Amazon Redshift targets, the data is truncated to 1024000 bytes.
		  - For Microsoft Azure Synapse Analytics targets, the data is truncated to 1000000 bytes.
		  - For Google BigQuery and Snowflake targets, the data is truncated to 8388608 bytes.
		- CLOB, DBCLOB, NCLOB, LONG, LONG VARCHAR, LONG VARGRAPHIC, TEXT, NTEXT, NVARCHAR(MAX), RAW, VARCHAR(MAX), or XML columns are truncated before being written to VARCHAR columns on the target.
		  - For Amazon S3, Databricks, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, Oracle, Microsoft Fabric OneLake, Oracle Cloud Object Storage, PostgreSQL, and Snowflake targets, the data is truncated to 16777216 bytes.
		  - For Amazon Redshift targets, the data is truncated to 65535 bytes.

Property	Source and Load Type	Description
	Snowflake, or SQL Server targets. - Incremental loads to Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka-enabled Azure Event Hubs, Microsoft Azure Data Lake Storage Gen 2, Microsoft Azure Synapse Analytics, Oracle, Oracle Cloud Object Storage, Snowflake, or SQL Server targets. - Combined initial and incremental loads to Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka-enabled Azure Event Hubs, Microsoft Azure Data Lake Storage Gen 2, Microsoft AzureSynapse Analytics, Oracle, Oracle Cloud Object Storage, Snowflake, or SQL Server targets. - PostgreSQL LOBs are not supported with PostgreSQL targets. SQL Server sources: - Initial loads to Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen 2, Microsoft Azure Synapse Analytics, Oracle, Oracle Cloud Object Storage, Snowflake, or SQL Server targets. - Incremental loads to Kafka-enabled Azure Event Hubs, Databricks, Snowflake, and SQL Server targets. Disabled if you selected the **Query-based** CDC method. - Combined initial and incremental loads to Databricks, Snowflake, and SQL Server targets. Disabled if you selected the **Query-based** CDC method. - SQL Server LOBs are not supported with PostgreSQL targets.	- For Google BigQuery targets, the data is truncated to 8388608 bytes. - For Microsoft Azure Synapse Analytics targets, the data is truncated to 500000 bytes. - For SQL Server targets, CLOB, TEXT and VARCHAR(MAX) data is truncated to 16777216 bytes, NCLOB, NTEXT and NVARCHAR(MAX) data to 33554432 bytes, and XML data to 33554442 bytes. - For Azure Event Hubs targets, the overall record maximum size is 1 MB. If the record size exceeds 1 MB, Azure Event Hubs generates an error, and the task fails. Source-side truncation considerations: - For Db2 for LUW and Oracle sources, you can specify the custom properties dbmiSourceBlobTruncationSize and dbmiSourceClobTruncationSize on the Source page to control the number of bytes at which truncation occurs for blob and clob types of data, respectively, when you want the data truncated at a point less than the maximum size that the target allows. - For PostgreSQL sources in incremental loads and combined loads, if large-object columns contain more than 1 MB of data, the data is truncated to 1 MB.

Property	Source and Load Type	Description
Enable Persistent Storage	All sources except Db2 for LUW (query-based CDC), MongoDB, Oracle (query-based CDC), PostgreSQL, SAP HANA, SAP HANA Cloud, and SQL Server (query-based CDC) - Incremental loads and combined initial and incremental loads. For Db2 for LUW, Oracle, and SQL Server sources that use the query-based CDC method, this field is not displayed because persistent storage is enabled by default and cannot be changed. For MongoDB, PostgreSQL, SAP HANA, and SAP HANA Cloud change data sources, the field is not displayed because persistent storage is enabled by default and cannot be changed.	Select this check box to enable persistent storage of transaction data in a disk buffer so that the data can be consumed continually, even when the writing of data to the target is slow or delayed. Benefits of using persistent storage are faster consumption of the source transaction logs, less reliance on log archives or backups, and the ability to still access the data persisted in disk storage after restarting a database ingestion job.
Enable Partitioning	Oracle sources - Initial loads and combined initial and incremental loads SQL Server sources - Initial loads and combined initial and incremental loads	Select this check box to enable partitioning of source objects. When an object is partitioned, the database ingestion and replication job processes the records read from each partition in parallel. For Oracle sources, database ingestion and replication determines the range of partitions by using the ROWID as the partition key. Also, when you select the **Enable Partitioning** check box, the **Disable Flashback** check box is automatically selected. For SQL Server sources, partitioning is based on the primary key. **Note:** In combined initial and incremental loads, the partitioning of source objects occurs only in the initial load phase.
Number of Partitions	Oracle sources - Initial loads and combined initial and incremental loads SQL Server sources - Initial loads and combined initial and incremental loads	If you enable partitioning of source objects, enter the number of partitions you want to create. The default number is 5. The minimum value is 2.

Property	Source and Load Type	Description

Initial Start Point for Incremental Load	All sources - Incremental loads	Set this field if you want to customize the position in the source logs from which the database ingestion and replication job starts reading change records the first time it runs.
		Options are:
		- **Earliest Available**. The earliest available position in the database log or structure where changes are stored.
		  - For Db2 for i, the start of the current journal.
		  - For Db2 for LUW, this option is not available.
		  - For Db2 for z/OS, the earliest available record in the transaction log.
		  - For MongoDB, this option is not available.
		  - For MySQL, the earliest available record in the first binlog file.
		  - For Oracle, this option is not available.
		  - For PostgreSQL, the earliest available record in the replication slot.
		  - For SAP HANA and SAP HANA Cloud, the earliest available record in the PKLOG table.
		  - For SQL Server, the earliest available record in the active transaction log. This option is not available if the **CDC Method** is set to **Query-based**.
		- **Latest Available**. The latest available position in the database log or structure.
		- **Position**. A position in the change stream from which you want the database ingestion job to start retrieving change records. The position value is a Db2 for i timestamp, Db2 for z/OS LRSN, Oracle SCN, PostgreSQL LSN, SAP HANA sequence value, or SQL Server LSN. This value must be equal to or less than the current position value. An invalid value will cause the job to fail. The value of 0 is displayed by default, which results in the following start positions:
		  - For Db2 for i, do not use the default value of 0.
		  - For Db2 for LUW, this option is not available.
		  - For Db2 for z/OS, the value of 0 causes the latest available point to be used.
		  - For MongoDB and MySQL, this option is not available.
		  - For Oracle, the value of 0 causes the latest available point to be used. This option is not available if the **CDC Method** is set to **Query-based**.
		  - For PostgreSQL, the value of 0 causes the earliest available point to be used.
		  - For SAP HANA, the value of 0 causes the earliest available point to be used.
		  - For SQL Server, the value of 0 causes the earliest available point to be used. A non-zero LSN that predates the beginning of the active transaction log causes data to be read from the CDC tables instead of from the transaction log. This option is not available if the **CDC Method** is set to **Query-based**.

Property	Source and Load Type	Description
		- **Specific Date and Time**. A date and time, in the format MM/DD/YYYY hh:mm AM|PM, that Database Ingestion and Replication uses to determine the position in the change stream from which to start retrieving change records. Database Ingestion and Replication retrieves only the changes that were started after this date and time. If you enter a date and time earlier than the earliest date and time in the available archived logs, the job will fail. For MySQL sources, this option is not available. The default is **Latest Available**. **Notes:** - The **Initial Start Point for Incremental Load** option pertains only to the initial run of a job. Thereafter, if you resume a stopped or aborted job, the job begins propagating source data from where it last left off. - For SQL Server incremental load jobs that use log-based CDC with the transaction log, data changes are read from the active portion of the transaction log if the requested LSN is available there. If the LSN predates the active transaction log, the data changes are read from previously enabled CDC tables. Ensure that SQL Server CDC is enabled on the source tables. - For combined initial and incremental load jobs, initial loading is not performed until the incremental processing of change data reaches the end of the current transaction log.
Fetch Size	MongoDB - Initial loads and incremental loads	For a MongoDB source, the number of records that a database ingestion and replication job must read at a single time from the source. Valid values are 1 to 2147483647. The default is 5000.

15. Under **Custom Properties**, you can specify custom properties that Informatica provides to meet your special requirements. To add a property, in the **Create Property** fields, enter the property name and value. Then click **Add Property**.

    Specify these properties only at the direction of Informatica Global Customer Support. Usually, these properties address unique environments or special processing needs. You can specify multiple properties, if necessary. A property name can contain only alphanumeric characters and the following special characters: periods (.), hyphens (-), and underscores (_).

    Tip: To delete a property, click the Delete icon at the right end of the property row in the list.

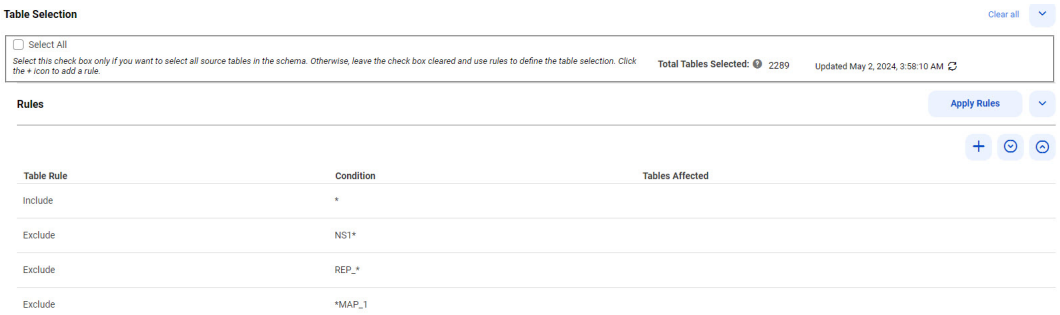
16. Click **Next**.

Example of rules for selecting source tables

When you define a source for a database ingestion and replication task, you can optionally define table selection rules to select a subset of the source tables in the specified schema. This simple example demonstrates how to use selection rules to select the tables you want.

Assume that 2984 tables are in the source schema. You want to exclude the tables from which you do not need to replicate data.

Define the following rules in the order shown:



The rules are processed from top to bottom.

- Rule 1 includes all source tables in the schema.
- Rule 2 excludes the source tables that have names beginning with "NS1".
- Rule 3 excludes the source tables that have names beginning with "REP_".
- Rule 4 excludes the source tables that end with "MAP_1".

After clicking the Refresh icon, the **Tables Selected** field shows 2289 tables, which indicates you filtered out 695 tables.

Configuring the target

Configure the target on the **Target** page of the database ingestion and replication task wizard.

1. In the **Connection** list, select a connection for the target type. The connection type appears in parentheses after the connection name.

   You must have previously defined the connection in Administrator for the runtime environment.

   The list includes only the connection types that are valid for the load type selected on the **Definition** page. No connections are listed if you did not select a load type.

   If you change the load type and the selected connection is no longer valid, a warning message is issued and the **Connection** field is cleared. You must select another connection that is valid for the updated load type.

   Note: After you deploy the database ingestion and replication task, you cannot change the connection without first undeploying the associated job. You must then deploy the task again.

2. In the **Target** section, configure the properties that pertain to your target type.

   For descriptions of these properties, see the following topics:

   - “Amazon Redshift target properties ” on page 122
   - “Amazon S3 target properties” on page 123
   - “Databricks target properties” on page 127
   - “Flat File target properties” on page 129
   - “Google BigQuery target properties” on page 132
   - “Google Cloud Storage target properties” on page 134
   - “Kafka target properties” on page 138
   - “Microsoft Azure Data Lake Storage target properties” on page 141
   - “Microsoft Azure Synapse Analytics target properties” on page 145
   - “Microsoft Fabric OneLake target properties” on page 146

- "Microsoft SQL Server target properties" on page 149
- "Oracle target properties" on page 150
- "Oracle Cloud Object Storage target properties" on page 151
- "PostgreSQL target properties" on page 155
- "Snowflake Data Cloud target properties" on page 155

3. If you want to rename the target objects that are associated with the selected source tables, define table renaming rules.

   For example, you can add a prefix such as TGT_. For more information, see "Renaming tables on the target" on page 119.

4. If you want to override the default mappings of source data types to target data types, define data type rules.

   This feature is supported only for tasks that have an Oracle (any load type) source and an SQL-based target type. For more information, see "Customizing data type mappings" on page 120.

5. Under **Custom Properties**, you can specify custom properties that Informatica provides to meet your special requirements. To add a property, in the **Create Property** fields, enter the property name and value. Then click **Add Property**.

   Specify these properties only at the direction of Informatica Global Customer Support. Usually, these properties address unique environments or special processing needs. You can specify multiple properties, if necessary. A property name can contain only alphanumeric characters and the following special characters: periods (.), hyphens (-), and underscores (_).

   Tip: To delete a property, click the Delete icon button at the right end of the property row in the list.

6. Click **Next** if available, or click **Save**.

Renaming tables on the target

When you configure a target with an existing schema, you can optionally define rules for renaming the target tables that correspond to the selected source tables.

For target messaging systems, such as Apache Kafka, the rule renames the table name in the output messages.

To create a rule for renaming tables:

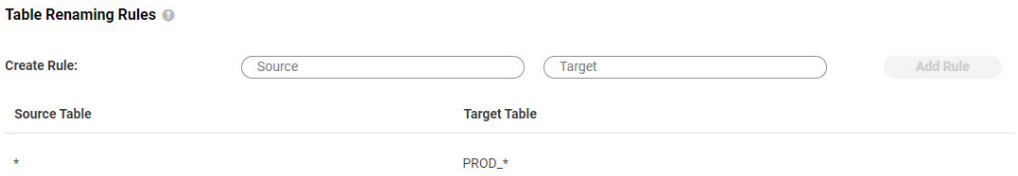
1. Under **Table Renaming Rules**, in the **Create Rule** fields, enter a source table name or a table name mask that includes one or more wildcards. Then enter the corresponding target table name or table name mask.

   Notes:

   - For the source, you can enter only the asterisk (*) wildcard to select all source tables that match the table selection criteria on the **Source** page. Or you can enter a specific source table name or a table-name mask that includes one or more of the following wildcards: an asterisk (*) to represent one or more characters or a question mark (?) to represent a single character.
   - To use a table-name mask with the wildcard character for the target, you must also use the wildcard character in the source. If you use a specific source table name with a target table mask that includes the wildcard character, the task deployment will fail.
   - If a table name includes special characters, such as a backslash (\), asterisk(*), dot (.), or question mark (?), escape each special character in the name with a backslash (\).

- On Windows, if you enter target table renaming criteria that causes a target table name to exceed 232 characters in length, the name is truncated to 222 characters. Database Ingestion and Replication appends 14 characters to the name to add a date-time yyyyMMddHHmmss value, which causes the name to exceed the Windows maximum limit of 255. Ensure that the names of any renamed target tables will not exceed 232 characters.

2. Click **Add Rule**.
   The rule appears in the rules list.



You can define multiple table rules. The order of the rules does not matter with regard to how they are processed unless a table matches multiple rules. In this case, the last matching rule determines the name of the table.

To delete a rule, click the Delete icon at the right end of the rule row.

Example:

Assume that you want to add the prefix "PROD_" to the names of target tables that correspond to all selected source tables. Enter the following values:

- For the source, enter only the asterisk (*) wildcard character to specify all of the selected source tables.
- For the target, enter PROD_* to add this prefix to the names of all target tables that match the source tables by name.

Customizing data type mappings

When you configure a target for a database ingestion and replication task, you can optionally define data-type mapping rules to override the default mappings of source data types to target data types.

The default mappings are described in "Default Data Type Mappings" on page 172.

This feature is supported for tasks that have the following source and target combinations:

- Oracle sources with Databricks, Google BigQuery, Microsoft Azure Synapse Analytics, Oracle, SQL Server, and Snowflake targets
- SQL Server sources with Snowflake targets

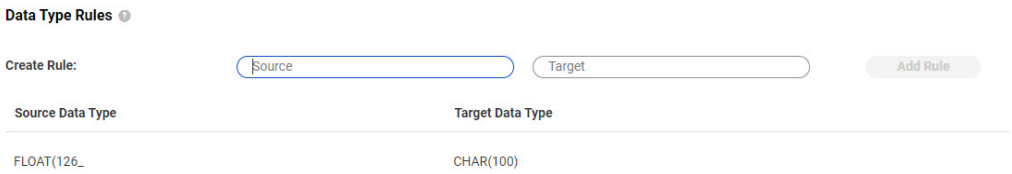
For example, you can create a data-type rule that maps Oracle NUMBER columns that have no precision to Snowflake target NUMBER() columns that also have no precision, instead of using the default mapping to the Snowflake VARCHAR(255) data type.

To create a data-type mapping rule:

1. Expand **Data Type Rules**.
2. In the **Create Rule** fields, enter a source data type and the target data type that you want to map it to. In the *Source* field only, you can include the percent (%) wildcard to represent the data type precision, scale, or size, for example, NUMBER(%,4), NUMBER(8,%), or NUMBER(%). Use the wildcard to cover all source columns that have the same data type but use different precision, scale, or size values, instead of specifying each one individually. For example, enter FLOAT(%) to cover FLOAT(16), FLOAT(32), and FLOAT(84). You cannot enter the % wildcard in the target data type. A source data type that uses the % wildcard must map to a target data type that uses specific precision, scale, or size value. For example,

you could map the source data type FLOAT(%) to a target data type specification such as NUMBER(38,10).

3. Click **Add Rule**.
   The rule appears in the list of rules.



To delete a rule, click the Delete icon at the right end of the rule row.

After you deploy a task with custom mapping rules, you cannot edit the rules until the task is undeployed.

Usage notes:

- In general, binary data types cannot be mapped to character data types.
- If you define multiple data-type rules for the same source data type with the same length or same precision and scale values, you will not be able to save the database ingestion and replication task.
- If you define multiple data-type rules for the same source data type but use the % wildcard to represent the length or precision and scale value in one rule and a specific length or precision and scale value in the second rule, the rule that contains the specific value is processed first, before the rule with the % wildcard. For example, if you map the source data types FLOAT(84) and FLOAT(%), the FLOAT(84) rule is processed first and then the FLOAT(%) rule is processed to cover any other FLOAT source columns with different sizes.
- If a source data type requires a length or precision and scale value, make sure that you set the required attribute by using the % wildcard or a specific value, for example, VARCHAR(%) or VARCHAR(10).
- If you define an invalid mapping, an error message is written to the log. You can then correct the mapping error, with assistance from your DBA if necessary.
- For Oracle sources, you must use the data types that are returned by the following query for the source object:

  ```
  select dbms_metadata.get_ddl('TABLE', 'YOUR_TABLE_NAME','TABLE_OWNER_NAME') from dual;
  ```

- Database Ingestion and Replication does not support the BYTE and CHAR semantics in data-type mappings rules.
- If a source data type has a default precision, you must specify it in your rule. For example, you must use TIMESTAMP(6) instead of TIMESTAMP.

Amazon Redshift target properties

When you define a database ingestion and replication task that has an Amazon Redshift target, you must enter some target properties on the **Target** tab of the task wizard.

The following table describes the Amazon Redshift target properties that appear under **Target**:

Property	Description
Target Creation	The only available option is **Create Target Tables**, which generates the target tables based on the source tables. **Note:** After the target table is created, Database Ingestion and Replication intelligently handles the target tables on subsequent job runs. Database Ingestion and Replication might truncate or re-create the target tables depending on the specific circumstances.
Schema	Select the target schema in which Database Ingestion and Replication creates the target tables.
Bucket	Specifies the name of the Amazon S3 bucket that stores, organizes, and controls access to the data objects that you load to Amazon Redshift.
Data Directory or Task Target Directory	Specifies the subdirectory where Database Ingestion and Replication stores output files for jobs associated with the task. This field is called **Data Directory** for an initial load job or **Task Target Directory** for an incremental load or combined initial and incremental load job.

The following table describes advanced target properties that appear under **Advanced**:

Property	Description
Enable Case Transformation	By default, target table names and column names are generated in the same case as the corresponding source names, unless cluster-level or session-level properties on the target override this case-sensitive behavior. If you want to control the case of letters in the target names, select this check box. Then select a **Case Transformation Strategy** option.
Case Transformation Strategy	If you selected **Enable Case Transformation**, select one of the following options to specify how to handle the case of letters in generated target table (or object) names and column (or field) names: - **Same as source**. Use the same case as the source table (or object) names and column (or field) names. - **UPPERCASE**. Use all uppercase. - **lowercase**. Use all lowercase. The default value is **Same as source**. **Note:** The selected strategy will override any cluster-level or session-level properties on the target for controlling case.

Amazon S3 target properties

When you define a database ingestion and replication task that has an Amazon S3 target, you must enter some target properties on the **Target** tab of the task wizard.

Under **Target**, you can enter the following Amazon S3 target properties:

Property	Description
Output Format	Select the format of the output file. Options are: - **CSV** - **AVRO** - **PARQUET** The default value is **CSV**. **Note:** Output files in CSV format use double-quotation marks ("") as the delimiter for each field.
Add Headers to CSV File	If **CSV** is selected as the output format, select this check box to add a header with source column names to the output CSV file.
Avro Format	If you selected **AVRO** as the output format, select the format of the Avro schema that will be created for each source table. Options are: - **Avro-Flat**. This Avro schema format lists all Avro fields in one record. - **Avro-Generic**. This Avro schema format lists all columns from a source table in a single array of Avro fields. - **Avro-Nested**. This Avro schema format organizes each type of information in a separate record. The default value is **Avro-Flat**.
Avro Serialization Format	If **AVRO** is selected as the output format, select the serialization format of the Avro output file. Options are: - **None** - **Binary** - **JSON** The default value is **Binary**.
Avro Schema Directory	If **AVRO** is selected as the output format, specify the local directory where Database Ingestion and Replication stores Avro schema definitions for each source table. Schema definition files have the following naming pattern: *schemaname_tablename*.txt **Note:** If this directory is not specified, no Avro schema definition file is produced.
File Compression Type	Select a file compression type for output files in CSV or AVRO output format. Options are: - **None** - **Deflate** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Avro Compression Type	If **AVRO** is selected as the output format, select an Avro compression type. Options are: - **None** - **Bzip2** - **Deflate** - **Snappy** The default value is **None**, which means no compression is used.

Property	Description
Parquet Compression Type	If the **PARQUET** output format is selected, you can select a compression type that is supported by Parquet. Options are: - **None** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Deflate Compression Level	If **Deflate** is selected in the **Avro Compression Type** field, specify a compression level from 0 to 9. The default value is 0.
Add Directory Tags	For incremental load and combined initial and incremental load tasks, select this check box to add the "dt=" prefix to the names of apply cycle directories to be compatible with the naming convention for Hive partitioning. This check box is cleared by default.
Task Target Directory	For incremental load and combined initial and incremental load tasks, the root directory for the other directories that hold output data files, schema files, and CDC cycle contents and completed files. You can use it to specify a custom root directory for the task. If you enable the **Connection Directory as Parent** option, you can still optionally specify a task target directory to use with the parent directory specified in the connection properties. This field is required if the {TaskTargetDirectory} placeholder is specified in patterns for any of the following directory fields.
Connection Directory as Parent	Select this check box to use the directory value that is specified in the target connection properties as the parent directory for the custom directory paths specified in the task target properties. For initial load tasks, the parent directory is used in the **Data Directory** and **Schema Directory**. For incremental load and combined initial and incremental load tasks, the parent directory is used in the **Data Directory**, **Schema Directory**, **Cycle Completion Directory**, and **Cycle Contents Directory**. This check box is selected by default. If you clear it, for initial loads, define the full path to the output files in the **Data Directory** field. For incremental loads, optionally specify a root directory for the task in the **Task Target Directory**.

Property	Description

Data Directory	*For initial load tasks*, define a directory structure for the directories where Database Ingestion and Replication stores output data files and optionally stores the schema. To define directory pattern, you can use the following types of entries:
	- The placeholders {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the initial load job starts to transfer data to the target.
	- Specific directory names.
	- The toUpper() and toLower() functions, which force the values for an associated (*placeholder*) to uppercase or lowercase.
	Note: Placeholder values are not case sensitive.
	Examples:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	The default directory pattern is `{TableName)_{Timestamp}`.
	For incremental load and combined initial and incremental load tasks, define a custom path to the subdirectory that contains the cdc-data data files. To define the directory pattern, you can use the following types of entries:
	- The placeholders {TaskTargetDirectory}, {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the CDC cycle started.
	If you include the toUpper or toLower function, put the placeholder name in parentheses and enclose the both the function and placeholder in curly brackets, as shown in the preceding example.
	- Specific directory names.
	The default directory pattern is `{TaskTargetDirectory}/data/{TableName}/data`
	Note: For Amazon S3, Flat File, Microsoft Azure Data Lake Storage Gen2, and Oracle Cloud Object Store targets, Database Ingestion and Replication uses the directory specified in the target connection properties as the root for the data directory path when **Connection Directory as Parent** is selected. For Google Cloud Storage targets, Database Ingestion and Replication uses the **Bucket** name that you specify in the target properties for the ingestion task. For Microsoft Fabric OneLake targets, the parent directory is the path specified in the **Lakehouse Path** field in the Microsoft Fabric OneLake connection properties.

Schema Directory	Specify a custom directory in which to store the schema file if you want to store it in a directory other than the default directory. For initial loads, previously used values if available are shown in a drop-down list for your convenience. This field is optional. For initial loads, the schema is stored in the data directory by default. For incremental loads and combined initial and incremental loads, the default directory for the schema file is `{TaskTargetDirectory}/data/{TableName}/schema` You can use the same placeholders as for the **Data Directory** field. Ensure that you enclose placeholders with curly brackets { }. If you include the toUpper or toLower function, put the placeholder name in parentheses and enclose the both the function and placeholder in curly brackets, for example: `{toLower(SchemaName)}` **Note:** Schema is written only to output data files in CSV format. Data files in Parquet and Avro formats contain their own embedded schema.
Cycle Completion Directory	For incremental load and combined initial and incremental load tasks, the path to the directory that contains the cycle completed file. Default is `{TaskTargetDirectory}/cycle/completed`.

Property	Description
Cycle Contents Directory	For incremental load and combined initial and incremental load tasks, the path to the directory that contains the cycle contents files. Default is `{TaskTargetDirectory}/cycle/contents`.
Use Cycle Partitioning for Data Directory	For incremental load and combined initial and incremental load tasks, causes a timestamp subdirectory to be created for each CDC cycle, under each data directory. If this option is not selected, individual data files are written to the same directory without a timestamp, unless you define an alternative directory structure.
Use Cycle Partitioning for Summary Directories	For incremental load and combined initial and incremental load tasks, causes a timestamp subdirectory to be created for each CDC cycle, under the summary contents and completed subdirectories.
List Individual Files in Contents	For incremental load and combined initial and incremental load tasks, lists individual data files under the contents subdirectory. If **Use Cycle Partitioning for Summary Directories** is cleared, this option is selected by default. All of the individual files are listed in the contents subdirectory unless you can configure custom subdirectories by using the placeholders, such as for timestamp or date. If **Use Cycle Partitioning for Data Directory** is selected, you can still optionally select this check box to list individual files and group them by CDC cycle.

Under **Advanced**, you can enter the following Amazon S3 advanced target properties, which primarily apply to incremental loads:

Field	Description
Add Operation Type	Select this check box to add a metadata column that records the source SQL operation type in the output that the job propagates to the target. For incremental loads, the job writes "I" for insert, "U" for update, or "D" for delete. For initial loads, the job always writes "I" for insert. By default, this check box is selected for incremental load and initial and incremental load jobs, and cleared for initial load jobs.
Add Operation Time	Select this check box to add a metadata column that records the source SQL operation timestamp in the output that the job propagates to the target. For initial loads, the job always writes the current date and time. By default, this check box is not selected.
Add Operation Owner	Select this check box to add a metadata column that records the owner of the source SQL operation in the output that the job propagates to the target. For initial loads, the job always writes "INFA" as the owner. By default, this check box is not selected. This property is not available for jobs that have a MongoDB or PostgreSQL source. **Note:** This property is not supported for jobs that have a SQL Server source and use the CDC Tables capture method.

Field	Description
Add Operation Transaction Id	Select this check box to add a metadata column that includes the source transaction ID in the output that the job propagates to the target for SQL operations. For initial loads, the job always writes "1" as the ID. By default, this check box is not selected.
Add Before Images	Select this check box to include UNDO data in the output that a job writes to the target. For initial loads, the job writes nulls. By default, this check box is not selected.

Databricks target properties

When you define a database ingestion and replication task that has a Databricks target, you must enter some target properties on the **Target** tab of the task wizard.

The following table describes the Databricks target properties that appear under **Target**:

Property	Description
Target Creation	The only available option is **Create Target Tables**, which generates the target tables based on the source tables. **Note:** After the target table is created, Database Ingestion and Replication intelligently handles the target tables on subsequent job runs. Database Ingestion and Replication might truncate or re-create the target tables depending on the specific circumstances.
Schema	Select the target schema in which Database Ingestion and Replication creates the target tables.

Property	Description
Apply Mode	For incremental load and combined initial and incremental load jobs, indicates how source DML changes, including inserts, updates, and deletes, are applied to the target. Options are: - **Standard**. Accumulate the changes in a single apply cycle and intelligently merge them into fewer SQL statements before applying them to the target. For example, if an update followed by a delete occurs on the source row, no row is applied to the target. If multiple updates occur on the same column or field, only the last update is applied to the target. If multiple updates occur on different columns or fields, the updates are merged into a single update record before being applied to the target. - **Soft Deletes**. Apply source delete operations to the target as soft deletes. A soft delete marks the deleted row as deleted without actually removing it from the database. For example, a delete on the source results in a change record on the target with "D" displayed in the INFA_OPERATION_TYPE column. Consider using soft deletes if you have a long-running business process that needs the soft-deleted data to finish processing, to restore data after an accidental delete operation, or to track deleted values for audit purposes. **Note:** If you use **Soft Deletes** mode, you must not perform an update on the primary key in a source table. Otherwise, data corruption can occur on the target. - **Audit**. Apply an audit trail of every DML operation made on the source tables to the target. A row for each DML change on a source table is written to the generated target table along with the audit columns you select under the **Advanced** section. The audit columns contain metadata about the change, such as the DML operation type, time, owner, transaction ID, generated ascending sequence number, and before image. Consider using Audit apply mode when you want to use the audit history to perform downstream computations or processing on the data before writing it to the target database or when you want to examine metadata about the captured changes. The default value is **Standard**. **Note:** This field does not appear if you selected **Query-based** as the CDC method on the **Source** page of the task wizard.
Data Directory or Task Target Directory	Specifies the subdirectory where Database Ingestion and Replication stores output files for jobs associated with the task. This field is called **Data Directory** for an initial load job or **Task Target Directory** for an incremental load or combined initial and incremental load job.

The following table describes advanced target properties that appear under **Advanced**:

Property	Description
Add Operation Type	Select this check box to add a metadata column that records the source SQL operation type in the output that the job propagates to the target database or inserts into the audit table on the target system. This field is available only when the **Apply Mode** option is set to **Audit** or **Soft Deletes**. In Audit mode, the job writes "I" for inserts, "U" for updates, "E" for upserts, or "D" for deletes to this metadata column. In Soft Deletes mode, the job writes "D" for deletes or NULL for inserts and updates. When the operation type is NULL, the other "Add Operation..." metadata columns are also NULL. Only when the operation type is "D" will the other metadata columns contain non-null values. By default, this check box is selected. You cannot deselect it if you are using soft deletes.
Add Operation Time	Select this check box to add a metadata column that records the source SQL operation timestamp in the output that the job propagates to the target database or inserts into the audit table on the target system. This field is available only when **Apply Mode** is set to **Audit** or **Soft Deletes**. By default, this check box is not selected.

Property	Description
Add Operation Owner	Select this check box to add a metadata column that records the owner of the source SQL operation in the output that the job propagates to the target database or inserts into the audit table on the target system. This field is available only when **Apply Mode** is set to **Audit** or **Soft Deletes**. By default, this check box is not selected. This property is not available for jobs that have a MongoDB or PostgreSQL source. **Note:** This property is not supported for jobs that have a SQL Server source and use the CDC Tables capture method.
Add Operation Transaction Id	Select this check box to add a metadata column that includes the source transaction ID in the output that the job propagates to the target for SQL operations. This field is available only when **Apply Mode** is set to **Audit** or **Soft Deletes**. By default, this check box is not selected.
Add Operation Sequence	Select this check box to add a metadata column that records a generated, ascending sequence number for each change operation that the job inserts into the audit table on the target system. The sequence number reflects the change stream position of the operation. This field is available only when **Apply Mode** is set to **Audit**. By default, this check box is not selected.
Add Before Images	Select this check box to add _OLD columns with UNDO "before image" data in the output that the job inserts into the target tables. You can then compare the old and current values for each data column. For a delete operation, the current value will be null. This field is available only when **Apply Mode** is set to **Audit**. By default, this check box is not selected.
Prefix for Metadata Columns	Add a prefix to the names of the added metadata columns to easily identify them and to prevent conflicts with the names of existing columns. The default value is INFA_.
Create Unmanaged Tables	Select this check box if you want the task to create Databricks target tables as unmanaged tables. After you deploy the task, you cannot edit this field to switch to managed tables. By default, this option is cleared and managed tables are created. For more information about Databricks managed and unmanaged tables, see the Databricks documentation.
Unmanaged Tables Parent Directory	If you choose to create Databricks unmanaged tables, you must specify a parent directory in Amazon S3 or Microsoft Azure Data Lake Storage to hold the Parquet files that are generated for each target table when captured DML records are processed. **Note:** To use Unity Catalog, you must provide an existing external directory.

Flat File target properties

When you define a database ingestion and replication task, you must enter some properties for your Flat File target on the **Target** page of the task wizard.

Note: For flat file targets, these properties apply to initial load jobs only.

Under **Target**, you can enter the following Flat File target properties:

Property	Description
Output Format	Select the format of the output file. Options are: - **CSV** - **AVRO** The default value is **CSV**. **Note:** Output files in CSV format use double-quotation marks ("") as the delimiter for each field.
Add Headers to CSV File	If **CSV** is selected as the output format, select this check box to add a header with source column names to the output CSV file.
Avro Format	If you selected **AVRO** as the output format, select the format of the Avro schema that will be created for each source table. Options are: - **Avro-Flat**. This Avro schema format lists all Avro fields in one record. - **Avro-Generic**. This Avro schema format lists all columns from a source table in a single array of Avro fields. - **Avro-Nested**. This Avro schema format organizes each type of information in a separate record. The default value is **Avro-Flat**.
Avro Serialization Format	If **AVRO** is selected as the output format, select the serialization format of the Avro output file. Options are: - **None** - **Binary** - **JSON** The default value is **Binary**.
Avro Schema Directory	If **AVRO** is selected as the output format, specify the local directory where Database Ingestion and Replication stores Avro schema definitions for each source table. Schema definition files have the following naming pattern: *schemaname_tablename*.txt **Note:** If this directory is not specified, no Avro schema definition file is produced.
File Compression Type	Select a file compression type for output files in CSV or AVRO output format. Options are: - **None** - **Deflate** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Avro Compression Type	If **AVRO** is selected as the output format, select an Avro compression type. Options are: - **None** - **Bzip2** - **Deflate** - **Snappy** The default value is **None**, which means no compression is used.
Deflate Compression Level	If **Deflate** is selected in the **Avro Compression Type** field, specify a compression level from 0 to 9. The default value is 0.

Property	Description

Data Directory	For initial load tasks, define a directory structure for the directories where Database Ingestion and Replication stores output data files and optionally stores the schema. To define directory pattern, you can use the following types of entries:
	- The placeholders {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the initial load job starts to transfer data to the target.
	- Specific directory names.
	- The toUpper() and toLower() functions, which force the values for an associated (*placeholder*) to uppercase or lowercase.
	Note: Placeholder values are not case sensitive.
	Examples:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	The default directory pattern is `{TableName)_{Timestamp}`.
	Note: For Amazon S3, Flat File, and Microsoft Azure Data Lake Storage Gen2 targets, Database Ingestion and Replication uses the directory specified in the target connection properties as the root for the data directory path when **Connection Directory as Parent** is selected. For Google Cloud Storage targets, Database Ingestion and Replication uses the **Bucket** name that you specify in the target properties for the ingestion task.

Connection Directory as Parent	For initial load tasks, select this check box to use the directory value that is specified in the target connection properties as the parent directory for the custom directory paths specified in the task target properties. The parent directory is used in the **Data Directory** and **Schema Directory**.
Schema Directory	For initial load tasks, you can specify a custom directory in which to store the schema file if you want to store it in a directory other than the default directory. This field is optional. The schema is stored in the data directory by default. For incremental loads, the default directory for the schema file is `{TaskTargetDirectory}/data/{TableName}/schema`. You can use the same placeholders as for the **Data Directory** field. Ensure the placeholders are enclosed in curly brackets { }.

Under **Advanced**, you can enter the following advanced target properties:

Field	Description
Add Operation Type	Select this check box to add a metadata column that includes the source SQL operation type in the output that the job propagates to the target. For initial loads, the job always writes "I" for insert. By default, this check box is cleared.
Add Operation Time	Select this check box to add a metadata column that records the source SQL operation timestamp in the output that the job propagates to the target. For initial loads, the job always writes the current date and time. By default, this check box is not selected.

Field	Description
Add Operation Owner	Select this check box to add a metadata column that records the owner of the source SQL operation in the output that the job propagates to the target. For initial loads, the job always writes "INFA" as the owner. By default, this check box is not selected. This property is not available for jobs that have a MongoDB or PostgreSQL source. **Note:** This property is not supported for jobs that have a SQL Server source and use the CDC Tables capture method.
Add Operation Transaction Id	Select this check box to add a metadata column that includes the source transaction ID in the output that the job propagates to the target for SQL operations. For initial loads, the job always writes "1" as the ID. By default, this check box is not selected.
Add Before Images	Select this check box to include UNDO data in the output that a job writes to the target. For initial loads, the job writes nulls. By default, this check box is not selected.

Google BigQuery target properties

When you define a database ingestion and replication task that has a Google BigQuery target, you must enter some target properties on the **Target** tab of the task wizard.

The following table describes the Google BigQuery target properties that appear under **Target**:

Property	Description
Target Creation	The only available option is **Create Target Tables**, which generates the target tables based on the source tables. **Note:** After the target table is created, Database Ingestion and Replication intelligently handles the target tables on subsequent job runs. Database Ingestion and Replication might truncate or re-create the target tables depending on the specific circumstances.
Schema	Select the target schema in which Database Ingestion and Replication creates the target tables.
Apply Mode	For incremental load and combined initial and incremental load jobs, indicates how source DML changes, including inserts, updates, and deletes, are applied to the target. Options are: - **Standard**. Accumulate the changes in a single apply cycle and intelligently merge them into fewer SQL statements before applying them to the target. For example, if an update followed by a delete occurs on the source row, no row is applied to the target. If multiple updates occur on the same column or field, only the last update is applied to the target. If multiple updates occur on different columns or fields, the updates are merged into a single update record before being applied to the target. - **Audit**. Apply an audit trail of every DML operation made on the source tables to the target. A row for each DML change on a source table is written to the generated target table along with the audit columns you select under the **Advanced** section. The audit columns contain metadata about the change, such as the DML operation type, time, owner, transaction ID, generated ascending sequence number, and before image. Consider using Audit apply mode when you want to use the audit history to perform downstream computations or processing on the data before writing it to the target database or when you want to examine metadata about the captured changes. The default value is **Standard**. **Note:** This field does not appear if you selected **Query-based** as the CDC method on the **Source** page of the task wizard.

Property	Description
Bucket	Specifies the name of an existing bucket container that stores, organizes, and controls access to the data objects that you load to Google Cloud Storage.
Data Directory or Task Target Directory	Specifies the subdirectory where Database Ingestion and Replication stores output files for jobs associated with the task. This field is called **Data Directory** for an initial load job or **Task Target Directory** for an incremental load or combined initial and incremental load job.

The following table describes advanced target properties that appear under **Advanced**:

Property	Description
Add Last Replicated Time	Select this check box to add a metadata column that records the timestamp at which a record was inserted or last updated in the target table. For initial loads, all loaded records have the same timestamp. For incremental and combined initial and incremental loads, the column records the timestamp of the last DML operation that was applied to the target. By default, this check box is not selected.
Add Operation Type	Select this check box to add a metadata column that records the source SQL operation type in the output that the job propagates to the target tables. The job writes "I" for insert, "U" for update, or "D" for delete. This field is available only when **Apply Mode** is set to **Audit** By default, this check box is selected.
Add Operation Time	Select this check box to add a metadata column that records the source SQL operation timestamp in the output that the job propagates to the target tables. This field is available only when **Apply Mode** is set to **Audit**. By default, this check box is not selected.
Add Operation Owner	Select this check box to add a metadata column that records the owner of the source SQL operation in the output that the job propagates to the target tables. This field is available only when **Apply Mode** is set to **Audit**. By default, this check box is not selected. This property is not available for jobs that have a MongoDB or PostgreSQL source. **Note:** This property is not supported for jobs that have a SQL Server source and use the CDC Tables capture method.
Add Operation Transaction Id	Select this check box to add a metadata column that includes the source transaction ID in the output that the job propagates to the target for SQL operations. This field is available only when **Apply Mode** is set to **Audit**. By default, this check box is not selected.
Add Operation Sequence	Select this check box to add a metadata column that records a generated, ascending sequence number for each change operation that the job inserts into the target table. The sequence number reflects the change stream position of the operation. This field is available only when **Apply Mode** is set to **Audit**. By default, this check box is not selected.

Property	Description
Add Before Images	Select this check box to add _OLD columns with UNDO "before image" data in the output that the job inserts into the target table. You can then compare the old and current values for each data column. For a delete operation, the current value will be null. This field is available only when **Apply Mode** is set to **Audit**. By default, this check box is not selected.
Prefix for Metadata Columns	Add a prefix to the names of the added metadata columns to easily identify them and to prevent conflicts with the names of existing columns. Do not include special characters in the prefix. Otherwise, task deployment will fail. The default value is INFA_.
Enable Case Transformation	By default, target table names and column names are generated in the same case as the corresponding source names, unless cluster-level or session-level properties on the target override this case-sensitive behavior. If you want to control the case of letters in the target names, select this check box. Then select a **Case Transformation Strategy** option.
Case Transformation Strategy	If you selected **Enable Case Transformation**, select one of the following options to specify how to handle the case of letters in generated target table (or object) names and column (or field) names: - **Same as source**. Use the same case as the source table (or object) names and column (or field) names. - **UPPERCASE**. Use all uppercase. - **lowercase**. Use all lowercase. The default value is **Same as source**. **Note:** The selected strategy will override any cluster-level or session-level properties on the target for controlling case.

Google Cloud Storage target properties

When you define a database ingestion and replication task that has a Google Cloud Storage target, you must enter some target properties on the **Target** tab of the task wizard.

Under **Target**, you can enter the following Google Cloud Storage target properties:

Property	Description
Output Format	Select the format of the output file. Options are: - **CSV** - **AVRO** - **PARQUET** The default value is **CSV**. **Note:** Output files in CSV format use double-quotation marks ("") as the delimiter for each field.
Add Headers to CSV File	If **CSV** is selected as the output format, select this check box to add a header with source column names to the output CSV file.
Avro Format	If you selected **AVRO** as the output format, select the format of the Avro schema that will be created for each source table. Options are: - **Avro-Flat**. This Avro schema format lists all Avro fields in one record. - **Avro-Generic**. This Avro schema format lists all columns from a source table in a single array of Avro fields. - **Avro-Nested**. This Avro schema format organizes each type of information in a separate record. The default value is **Avro-Flat**.

Property	Description
Avro Serialization Format	If **AVRO** is selected as the output format, select the serialization format of the Avro output file. Options are: - **None** - **Binary** - **JSON** The default value is **Binary**.
Avro Schema Directory	If **AVRO** is selected as the output format, specify the local directory where Database Ingestion and Replication stores Avro schema definitions for each source table. Schema definition files have the following naming pattern: *schemaname_tablename*.txt **Note:** If this directory is not specified, no Avro schema definition file is produced.
File Compression Type	Select a file compression type for output files in CSV or AVRO output format. Options are: - **None** - **Deflate** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Avro Compression Type	If **AVRO** is selected as the output format, select an Avro compression type. Options are: - **None** - **Bzip2** - **Deflate** - **Snappy** The default value is **None**, which means no compression is used.
Parquet Compression Type	If the **PARQUET** output format is selected, you can select a compression type that is supported by Parquet. Options are: - **None** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Deflate Compression Level	If **Deflate** is selected in the **Avro Compression Type** field, specify a compression level from 0 to 9. The default value is 0.
Add Directory Tags	For incremental load and combined initial and incremental load tasks, select this check box to add the "dt=" prefix to the names of apply cycle directories to be compatible with the naming convention for Hive partitioning. This check box is cleared by default.
Bucket	Specifies the name of an existing bucket container that stores, organizes, and controls access to the data objects that you load to Google Cloud Storage.
Task Target Directory	For incremental load and combined initial and incremental load tasks, the root directory for the other directories that hold output data files, schema files, and CDC cycle contents and completed files. You can use it to specify a custom root directory for the task. If you enable the **Connection Directory as Parent** option, you can still optionally specify a task target directory to use with the parent directory specified in the connection properties. This field is required if the {TaskTargetDirectory} placeholder is specified in patterns for any of the following directory fields.

Property	Description

Data Directory	*For initial load tasks*, define a directory structure for the directories where Database Ingestion and Replication stores output data files and optionally stores the schema. To define directory pattern, you can use the following types of entries:
	- The placeholders {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the initial load job starts to transfer data to the target.
	- Specific directory names.
	- The toUpper() and toLower() functions, which force the values for an associated (*placeholder*) to uppercase or lowercase.
	Note: Placeholder values are not case sensitive.
	Examples:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	The default directory pattern is `{TableName)_{Timestamp}`.
	For incremental load and combined initial and incremental load tasks, define a custom path to the subdirectory that contains the cdc-data data files. To define the directory pattern, you can use the following types of entries:
	- The placeholders {TaskTargetDirectory}, {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the CDC cycle started.
	If you include the toUpper or toLower function, put the placeholder name in parentheses and enclose the both the function and placeholder in curly brackets, as shown in the preceding example.
	- Specific directory names.
	The default directory pattern is `{TaskTargetDirectory}/data/{TableName}/data`
	Note: For Amazon S3, Flat File, Microsoft Azure Data Lake Storage Gen2, and Oracle Cloud Object Store targets, Database Ingestion and Replication uses the directory specified in the target connection properties as the root for the data directory path when **Connection Directory as Parent** is selected. For Google Cloud Storage targets, Database Ingestion and Replication uses the **Bucket** name that you specify in the target properties for the ingestion task. For Microsoft Fabric OneLake targets, the parent directory is the path specified in the **Lakehouse Path** field in the Microsoft Fabric OneLake connection properties.

Schema Directory	Specify a custom directory in which to store the schema file if you want to store it in a directory other than the default directory. For initial loads, previously used values if available are shown in a drop-down list for your convenience. This field is optional. For initial loads, the schema is stored in the data directory by default. For incremental loads and combined initial and incremental loads, the default directory for the schema file is `{TaskTargetDirectory}/data/{TableName}/schema` You can use the same placeholders as for the **Data Directory** field. Ensure that you enclose placeholders with curly brackets { }. If you include the toUpper or toLower function, put the placeholder name in parentheses and enclose the both the function and placeholder in curly brackets, for example: `{toLower(SchemaName)}` **Note:** Schema is written only to output data files in CSV format. Data files in Parquet and Avro formats contain their own embedded schema.
Cycle Completion Directory	For incremental load and combined initial and incremental load tasks, the path to the directory that contains the cycle completed file. Default is `{TaskTargetDirectory}/cycle/completed`.

Property	Description
Cycle Contents Directory	For incremental load and combined initial and incremental load tasks, the path to the directory that contains the cycle contents files. Default is `{TaskTargetDirectory}/cycle/contents`.
Use Cycle Partitioning for Data Directory	For incremental load and combined initial and incremental load tasks, causes a timestamp subdirectory to be created for each CDC cycle, under each data directory. If this option is not selected, individual data files are written to the same directory without a timestamp, unless you define an alternative directory structure.
Use Cycle Partitioning for Summary Directories	For incremental load and combined initial and incremental load tasks, causes a timestamp subdirectory to be created for each CDC cycle, under the summary contents and completed subdirectories.
List Individual Files in Contents	For incremental load and combined initial and incremental load tasks, lists individual data files under the contents subdirectory. If **Use Cycle Partitioning for Summary Directories** is cleared, this option is selected by default. All of the individual files are listed in the contents subdirectory unless you can configure custom subdirectories by using the placeholders, such as for timestamp or date. If **Use Cycle Partitioning for Data Directory** is selected, you can still optionally select this check box to list individual files and group them by CDC cycle.

Under **Advanced**, you can enter the following Google Cloud Storage advanced target properties, which are primarily for incremental load jobs:

Field	Description
Add Operation Type	Select this check box to add a metadata column that records the source SQL operation type in the output that the job propagates to the target. For incremental loads, the job writes "I" for insert, "U" for update, or "D" for delete. For initial loads, the job always writes "I" for insert. By default, this check box is selected for incremental load and initial and incremental load jobs, and cleared for initial load jobs.
Add Operation Time	Select this check box to add a metadata column that records the source SQL operation timestamp in the output that the job propagates to the target. For initial loads, the job always writes the current date and time. By default, this check box is not selected.
Add Operation Owner	Select this check box to add a metadata column that records the owner of the source SQL operation in the output that the job propagates to the target. For initial loads, the job always writes "INFA" as the owner. By default, this check box is not selected. This property is not available for jobs that have a MongoDB or PostgreSQL source. **Note:** This property is not supported for jobs that have a SQL Server source and use the CDC Tables capture method.

Field	Description
Add Operation Transaction Id	Select this check box to add a metadata column that includes the source transaction ID in the output that the job propagates to the target for SQL operations. For initial loads, the job always writes "1" as the ID. By default, this check box is not selected.
Add Before Images	Select this check box to include UNDO data in the output that a job writes to the target. For initial loads, the job writes nulls. By default, this check box is not selected.

Kafka target properties

When you define a database ingestion and replication task, you must enter some properties for your Kafka target on the **Target** page of the task wizard.

These properties apply to incremental load operations only.

The following table describes the Kafka target properties that appear under **Target**:

Property	Description
Use Table Name as Topic Name	Indicates whether Database Ingestion and Replication writes messages that contain source data to separate topics, one for each source table, or writes all messages to a single topic. Select this check box to write messages to separate table-specific topics. The topic names match the source table names, unless you add the source schema name, a prefix, or a suffix in the **Include Schema Name**, **Table Prefix**, or **Table Suffix** properties. By default, this check box is cleared. With the default setting, you must specify the name of the single topic to which all messages are written in the **Topic Name** property.
Include Schema Name	When **Use Table Name as Topic Name** is selected, this check box appears and is selected by default. This setting adds the source schema name in the table-specific topic names. The topic names then have the format *schemaname_tablename*. If you do *not* want to include the schema name, clear this check box.
Table Prefix	When **Use Table Name as Topic Name** is selected, this property appears so that you can optionally enter a prefix to add to the table-specific topic names. For example, if you specify myprefix_, the topic names have the format myprefix_*tablename*. If you omit the underscore (_) after the prefix, the prefix is prepended to the table name.
Table Suffix	When **Use Table Name as Topic Name** is selected, this property appears so that you can optionally enter a suffix to add to the table-specific topic names. For example, if you specify _mysuffix, the topic names have the format *tablename*_mysuffix. If you omit the underscore (_) before the suffix, the suffix is appended to the table name.
Topic Name	If you do *not* select **Use table name as topic name**, you must enter the name of the single Kafka topic to which all messages that contain source data will be written.

Property	Description
Output Format	Select the format of the output file. Options are: - **CSV** - **AVRO** - **JSON** The default value is **CSV**. **Note:** Output files in CSV format use double-quotation marks ("") as the delimiter for each field. If your Kafka target uses Confluent Schema Registry to store schemas for incremental load jobs, you must select **AVRO** as the format.
JSON Format	If **JSON** is selected as the output format, select the level of detail of the output. Options are: - **Concise**. This format records only the most relevant data in the output, such as the operation type and the column names and values. - **Verbose**. This format records detailed information, such as the table name and column types.
Avro Format	If you selected **AVRO** as the output format, select the format of the Avro schema that will be created for each source table. Options are: - **Avro-Flat**. This Avro schema format lists all Avro fields in one record. - **Avro-Generic**. This Avro schema format lists all columns from a source table in a single array of Avro fields. - **Avro-Nested**. This Avro schema format organizes each type of information in a separate record. The default value is **Avro-Flat**.
Avro Serialization Format	If **AVRO** is selected as the output format, select the serialization format of the Avro output file. Options are: - **Binary** - **JSON** - **None** The default value is **Binary**. If you have a Confluent Kafka target that uses Confluent Schema Registry to store schemas, select **None**. Otherwise, Confluent Schema Registry does not register the schema. Do not select **None** if you are not using Confluent Scheme Registry.

Avro Schema Directory	If **AVRO** is selected as the output format, specify the local directory where Database Ingestion and Replication stores Avro schema definitions for each source table. Schema definition files have the following naming pattern:
	`schemaname_tablename.txt`
	Note: If this directory is not specified, no Avro schema definition file is produced.
	If a source schema change is expected to alter the target, the Avro schema definition file is regenerated with a unique name that includes a timestamp, in the following format:
	`schemaname_tablename_YYYYMMDDhhmmss.txt`
	This unique naming pattern ensures that older schema definition files are preserved for audit purposes.

Avro Compression Type	If **AVRO** is selected as the output format, select an Avro compression type. Options are: - **None** - **Bzip2** - **Deflate** - **Snappy** The default value is **None**, which means no compression is used.
Deflate Compression Level	If **Deflate** is selected in the **Avro Compression Type** field, specify a compression level from 0 to 9. The default value is 0.

Under **Advanced**, you can enter the following advanced target properties:

Property	Description
Add Operation Type	Select this check box to add a metadata column that includes the source SQL operation type in the output that the job propagates to the target. The job writes "I" for insert, "U" for update, or "D" for delete. By default, this check box is selected.
Add Operation Time	Select this check box to add a metadata column that records the source SQL operation timestamp in the output that the job propagates to the target. By default, this check box is not selected.
Add Operation Owner	Select this check box to add a metadata column that records the owner of the source SQL operation in the output that the job propagates to the target. By default, this check box is not selected. This property is not available for jobs that have a MongoDB or PostgreSQL source. **Note:** This property is not supported for jobs that have a SQL Server source and use the CDC Tables capture method.
Add Operation Transaction Id	Select this check box to add a metadata column that includes the source transaction ID in the output that the job propagates to the target for SQL operations. By default, this check box is not selected.
Add Before Images	Select this check box to include UNDO data in the output that a job writes to the target. By default, this check box is not selected.
Async Write	Controls whether to use synchronous delivery of messages to Kafka. - Clear this check box to use synchronous delivery. Kafka must acknowledge each message as received before Database Ingestion and Replication sends the next message. In this mode, Kafka is unlikely to receive duplicate messages. However, performance might be slower. - Select this check box to use asynchronous delivery. Database Ingestion and Replication sends messages as soon as possible, without regard for the order in which the changes were retrieved from the source. By default, this check box is selected.

Producer Configuration Properties	Specify a comma-separated list of *key=value* pairs to enter Kafka producer properties for Apache Kafka, Confluent Kafka, Amazon Managed Streaming for Apache Kafka (MSK), or Kafka-enabled Azure Event Hubs targets.
	If you have a Confluent target that uses Confluent Schema Registry to store schemas, you must specify the following properties:
	`schema.registry.url=url,`
	`key.serializer=org.apache.kafka.common.serialization.StringSerializer,`
	`value.serializer=io.confluent.kafka.serializers.KafkaAvroSerializer`
	You can specify Kafka producer properties in either this field or in the **Additional Connection Properties** field in the Kafka connection.
	If you enter the producer properties in this field, the properties pertain to the database ingestion jobs associated with this task only. If you enter the producer properties for the connection, the properties pertain to jobs for all tasks that use the connection definition, unless you override the connection-level properties for specific tasks by also specifying properties in the **Producer Configuration Properties** field.
	For information about Kafka producer properties, see the Apache Kafka, Confluent Kafka, Amazon MSK, or Azure Event Hubs documentation.

Microsoft Azure Data Lake Storage target properties

When you define a database ingestion and replication task that has a Microsoft Azure Data Lake Storage target, you must enter some target properties on the **Target** page of the task wizard.

Under **Target**, you can enter the following Microsoft Azure Data Lake Storage target properties:

Property	Description
Output Format	Select the format of the output file. Options are: - **CSV** - **AVRO** - **PARQUET** The default value is **CSV**. **Note:** Output files in CSV format use double-quotation marks ("") as the delimiter for each field.
Add Headers to CSV File	If **CSV** is selected as the output format, select this check box to add a header with source column names to the output CSV file.
Avro Format	If you selected **AVRO** as the output format, select the format of the Avro schema that will be created for each source table. Options are: - **Avro-Flat**. This Avro schema format lists all Avro fields in one record. - **Avro-Generic**. This Avro schema format lists all columns from a source table in a single array of Avro fields. - **Avro-Nested**. This Avro schema format organizes each type of information in a separate record. The default value is **Avro-Flat**.
Avro Serialization Format	If **AVRO** is selected as the output format, select the serialization format of the Avro output file. Options are: - **None** - **Binary** - **JSON** The default value is **Binary**.
Avro Schema Directory	If **AVRO** is selected as the output format, specify the local directory where Database Ingestion and Replication stores Avro schema definitions for each source table. Schema definition files have the following naming pattern: `schemaname_tablename.txt` **Note:** If this directory is not specified, no Avro schema definition file is produced.
File Compression Type	Select a file compression type for output files in CSV or AVRO output format. Options are: - **None** - **Deflate** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Avro Compression Type	If **AVRO** is selected as the output format, select an Avro compression type. Options are: - **None** - **Bzip2** - **Deflate** - **Snappy** The default value is **None**, which means no compression is used.

Property	Description
Parquet Compression Type	If the **PARQUET** output format is selected, you can select a compression type that is supported by Parquet. Options are: - **None** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Deflate Compression Level	If **Deflate** is selected in the **Avro Compression Type** field, specify a compression level from 0 to 9. The default value is 0.
Add Directory Tags	For incremental load and combined initial and incremental load tasks, select this check box to add the "dt=" prefix to the names of apply cycle directories to be compatible with the naming convention for Hive partitioning. This check box is cleared by default.
Task Target Directory	For incremental load and combined initial and incremental load tasks, the root directory for the other directories that hold output data files, schema files, and CDC cycle contents and completed files. You can use it to specify a custom root directory for the task. If you enable the **Connection Directory as Parent** option, you can still optionally specify a task target directory to use with the parent directory specified in the connection properties. This field is required if the {TaskTargetDirectory} placeholder is specified in patterns for any of the following directory fields.
Connection Directory as Parent	Select this check box to use the directory value that is specified in the target connection properties as the parent directory for the custom directory paths specified in the task target properties. For initial load tasks, the parent directory is used in the **Data Directory** and **Schema Directory**. For incremental load and combined initial and incremental load tasks, the parent directory is used in the **Data Directory**, **Schema Directory**, **Cycle Completion Directory**, and **Cycle Contents Directory**. This check box is selected by default. If you clear it, for initial loads, define the full path to the output files in the **Data Directory** field. For incremental loads, optionally specify a root directory for the task in the **Task Target Directory**.

Property	Description

Data Directory	*For initial load tasks*, define a directory structure for the directories where Database Ingestion and Replication stores output data files and optionally stores the schema. To define directory pattern, you can use the following types of entries:
	- The placeholders {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the initial load job starts to transfer data to the target.
	- Specific directory names.
	- The toUpper() and toLower() functions, which force the values for an associated (*placeholder*) to uppercase or lowercase.
	Note: Placeholder values are not case sensitive.
	Examples:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	The default directory pattern is `{TableName)_{Timestamp}`.
	For incremental load and combined initial and incremental load tasks, define a custom path to the subdirectory that contains the cdc-data data files. To define the directory pattern, you can use the following types of entries:
	- The placeholders {TaskTargetDirectory}, {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the CDC cycle started.
	If you include the toUpper or toLower function, put the placeholder name in parentheses and enclose the both the function and placeholder in curly brackets, as shown in the preceding example.
	- Specific directory names.
	The default directory pattern is `{TaskTargetDirectory}/data/{TableName}/data`
	Note: For Amazon S3, Flat File, Microsoft Azure Data Lake Storage Gen2, and Oracle Cloud Object Store targets, Database Ingestion and Replication uses the directory specified in the target connection properties as the root for the data directory path when **Connection Directory as Parent** is selected. For Google Cloud Storage targets, Database Ingestion and Replication uses the **Bucket** name that you specify in the target properties for the ingestion task. For Microsoft Fabric OneLake targets, the parent directory is the path specified in the **Lakehouse Path** field in the Microsoft Fabric OneLake connection properties.

Schema Directory	Specify a custom directory in which to store the schema file if you want to store it in a directory other than the default directory. For initial loads, previously used values if available are shown in a drop-down list for your convenience. This field is optional. For initial loads, the schema is stored in the data directory by default. For incremental loads and combined initial and incremental loads, the default directory for the schema file is `{TaskTargetDirectory}/data/{TableName}/schema` You can use the same placeholders as for the **Data Directory** field. Ensure that you enclose placeholders with curly brackets { }. If you include the toUpper or toLower function, put the placeholder name in parentheses and enclose the both the function and placeholder in curly brackets, for example: `{toLower(SchemaName)}` **Note:** Schema is written only to output data files in CSV format. Data files in Parquet and Avro formats contain their own embedded schema.
Cycle Completion Directory	For incremental load and combined initial and incremental load tasks, the path to the directory that contains the cycle completed file. Default is `{TaskTargetDirectory}/cycle/completed`.

Property	Description
Cycle Contents Directory	For incremental load and combined initial and incremental load tasks, the path to the directory that contains the cycle contents files. Default is `{TaskTargetDirectory}/cycle/contents`.
Use Cycle Partitioning for Data Directory	For incremental load and combined initial and incremental load tasks, causes a timestamp subdirectory to be created for each CDC cycle, under each data directory. If this option is not selected, individual data files are written to the same directory without a timestamp, unless you define an alternative directory structure.
Use Cycle Partitioning for Summary Directories	For incremental load and combined initial and incremental load tasks, causes a timestamp subdirectory to be created for each CDC cycle, under the summary contents and completed subdirectories.
List Individual Files in Contents	For incremental load and combined initial and incremental load tasks, lists individual data files under the contents subdirectory. If **Use Cycle Partitioning for Summary Directories** is cleared, this option is selected by default. All of the individual files are listed in the contents subdirectory unless you can configure custom subdirectories by using the placeholders, such as for timestamp or date. If **Use Cycle Partitioning for Data Directory** is selected, you can still optionally select this check box to list individual files and group them by CDC cycle.

Under **Advanced**, you can enter the following advanced target properties, which primarily pertain to incremental load and combined load jobs:

Field	Description
Add Operation Type	Select this check box to add a metadata column that records the source SQL operation type in the output that the job propagates to the target. For incremental loads, the job writes "I" for insert, "U" for update, or "D" for delete. For initial loads, the job always writes "I" for insert. By default, this check box is selected for incremental load and initial and incremental load jobs, and cleared for initial load jobs.
Add Operation Time	Select this check box to add a metadata column that records the source SQL operation timestamp in the output that the job propagates to the target. For initial loads, the job always writes the current date and time. By default, this check box is not selected.
Add Operation Owner	Select this check box to add a metadata column that records the owner of the source SQL operation in the output that the job propagates to the target. For initial loads, the job always writes "INFA" as the owner. By default, this check box is not selected. This property is not available for jobs that have a MongoDB or PostgreSQL source. **Note:** This property is not supported for jobs that have a SQL Server source and use the CDC Tables capture method.

Field	Description
Add Operation Transaction Id	Select this check box to add a metadata column that includes the source transaction ID in the output that the job propagates to the target for SQL operations. For initial loads, the job always writes "1" as the ID. By default, this check box is not selected.
Add Before Images	Select this check box to include UNDO data in the output that a job writes to the target. For initial loads, the job writes nulls. By default, this check box is not selected.

Microsoft Azure Synapse Analytics target properties

When you define a database ingestion and replication task, you must enter some properties for your Microsoft Azure Synapse Analytics target on the **Target** page of the task wizard.

These properties apply to initial load, incremental load, and combined initial and incremental load operations.

The following table describes the target properties that appear under **Target**:

Property	Description
Target Creation	The only available option is **Create Target Tables**, which generates the target tables based on the source tables. **Note:** After the target table is created, Database Ingestion and Replication intelligently handles the target tables on subsequent job runs. Database Ingestion and Replication might truncate or re-create the target tables depending on the specific circumstances.
Schema	Select the target schema in which Database Ingestion and Replication creates the target tables. The schema name that is specified in the connection properties is displayed by default. Because this field is case sensitive, ensure that you entered the schema name in the connection properties in the correct case.

The following table describes advanced target properties that appear under **Advanced**:

Property	Description
Add Last Replicated Time	Select this check box to add a metadata column that records the timestamp at which a record was inserted or last updated in the target table. For initial loads, all loaded records have the same timestamp. For incremental and combined initial and incremental loads, the column records the timestamp of the last DML operation that was applied to the target. By default, this check box is not selected.
Prefix for Metadata Columns	Add a prefix to the names of the added metadata columns to easily identify them and to prevent conflicts with the names of existing columns. Do not include special characters in the prefix. Otherwise, task deployment will fail. The default value is INFA_.

Microsoft Fabric OneLake target properties

When you define a database ingestion and replication task that has a Microsoft Fabric OneLake target, you must enter some target properties on the **Target** page of the task wizard.

Under **Target**, you can enter the following Microsoft Fabric OneLake target properties:

Property	Description
Output Format	Select the format of the output file. Options are: - **CSV** - **AVRO** - **PARQUET** The default value is **CSV**. **Note:** Output files in CSV format use double-quotation marks ("") as the delimiter for each field.
Add Headers to CSV File	If **CSV** is selected as the output format, select this check box to add a header with source column names to the output CSV file.
Avro Format	If you selected **AVRO** as the output format, select the format of the Avro schema that will be created for each source table. Options are: - **Avro-Flat**. This Avro schema format lists all Avro fields in one record. - **Avro-Generic**. This Avro schema format lists all columns from a source table in a single array of Avro fields. - **Avro-Nested**. This Avro schema format organizes each type of information in a separate record. The default value is **Avro-Flat**.
Avro Serialization Format	If **AVRO** is selected as the output format, select the serialization format of the Avro output file. Options are: - **None** - **Binary** - **JSON** The default value is **Binary**.
Avro Schema Directory	If **AVRO** is selected as the output format, specify the local directory where Database Ingestion and Replication stores Avro schema definitions for each source table. Schema definition files have the following naming pattern: *schemaname_tablename*.txt **Note:** If this directory is not specified, no Avro schema definition file is produced.
File Compression Type	Select a file compression type for output files in CSV or AVRO output format. Options are: - **None** - **Deflate** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Avro Compression Type	If **AVRO** is selected as the output format, select an Avro compression type. Options are: - **None** - **Bzip2** - **Deflate** - **Snappy** The default value is **None**, which means no compression is used.

Property	Description
Parquet Compression Type	If the **PARQUET** output format is selected, you can select a compression type that is supported by Parquet. Options are: - **None** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Deflate Compression Level	If **Deflate** is selected in the **Avro Compression Type** field, specify a compression level from 0 to 9. The default value is 0.
Add Directory Tags	For incremental load and combined initial and incremental load tasks, select this check box to add the "dt=" prefix to the names of apply cycle directories to be compatible with the naming convention for Hive partitioning. This check box is cleared by default.
Task Target Directory	For incremental load and combined initial and incremental load tasks, the root directory for the other directories that hold output data files, schema files, and CDC cycle contents and completed files. You can use it to specify a custom root directory for the task. This field is required if the {TaskTargetDirectory} placeholder is specified in patterns for any of the following directory fields.

Property	Description

Data Directory	*For initial load tasks*, define a directory structure for the directories where Database Ingestion and Replication stores output data files and optionally stores the schema. To define directory pattern, you can use the following types of entries:
	- The placeholders {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the initial load job starts to transfer data to the target.
	- Specific directory names.
	- The toUpper() and toLower() functions, which force the values for an associated (*placeholder*) to uppercase or lowercase.
	Note: Placeholder values are not case sensitive.
	Examples:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	The default directory pattern is `{TableName)_{Timestamp}`.
	For incremental load and combined initial and incremental load tasks, define a custom path to the subdirectory that contains the cdc-data data files. To define the directory pattern, you can use the following types of entries:
	- The placeholders {TaskTargetDirectory}, {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the CDC cycle started.
	If you include the toUpper or toLower function, put the placeholder name in parentheses and enclose the both the function and placeholder in curly brackets, as shown in the preceding example.
	- Specific directory names.
	The default directory pattern is `{TaskTargetDirectory}/data/{TableName}/data`
	Note: For Amazon S3, Flat File, Microsoft Azure Data Lake Storage Gen2, and Oracle Cloud Object Store targets, Database Ingestion and Replication uses the directory specified in the target connection properties as the root for the data directory path when **Connection Directory as Parent** is selected. For Google Cloud Storage targets, Database Ingestion and Replication uses the **Bucket** name that you specify in the target properties for the ingestion task. For Microsoft Fabric OneLake targets, the parent directory is the path specified in the **Lakehouse Path** field in the Microsoft Fabric OneLake connection properties.

Schema Directory	Specify a custom directory in which to store the schema file if you want to store it in a directory other than the default directory. For initial loads, previously used values if available are shown in a drop-down list for your convenience. This field is optional. For initial loads, the schema is stored in the data directory by default. For incremental loads and combined initial and incremental loads, the default directory for the schema file is `{TaskTargetDirectory}/data/{TableName}/schema` You can use the same placeholders as for the **Data Directory** field. Ensure that you enclose placeholders with curly brackets { }. If you include the toUpper or toLower function, put the placeholder name in parentheses and enclose the both the function and placeholder in curly brackets, for example: `{toLower(SchemaName)}` **Note:** Schema is written only to output data files in CSV format. Data files in Parquet and Avro formats contain their own embedded schema.
Cycle Completion Directory	For incremental load and combined initial and incremental load tasks, the path to the directory that contains the cycle completed file. Default is `{TaskTargetDirectory}/cycle/completed`.

Property	Description
Cycle Contents Directory	For incremental load and combined initial and incremental load tasks, the path to the directory that contains the cycle contents files. Default is `{TaskTargetDirectory}/cycle/contents`.
Use Cycle Partitioning for Data Directory	For incremental load and combined initial and incremental load tasks, causes a timestamp subdirectory to be created for each CDC cycle, under each data directory. If this option is not selected, individual data files are written to the same directory without a timestamp, unless you define an alternative directory structure.
Use Cycle Partitioning for Summary Directories	For incremental load and combined initial and incremental load tasks, causes a timestamp subdirectory to be created for each CDC cycle, under the summary contents and completed subdirectories.
List Individual Files in Contents	For incremental load and combined initial and incremental load tasks, lists individual data files under the contents subdirectory. If **Use Cycle Partitioning for Summary Directories** is cleared, this option is selected by default. All of the individual files are listed in the contents subdirectory unless you can configure custom subdirectories by using the placeholders, such as for timestamp or date. If **Use Cycle Partitioning for Data Directory** is selected, you can still optionally select this check box to list individual files and group them by CDC cycle.

Microsoft SQL Server target properties

When you define a database ingestion and replication task, you must enter some properties for your Microsoft SQL Server target on the **Target** page of the task wizard.

The following table describes the target properties that appear under **Target**:

Property	Description
Target Creation	The only available option is **Create Target Tables**, which generates the target tables based on the source tables. **Note:** After the target table is created, Database Ingestion and Replication intelligently handles the target tables on subsequent job runs. Database Ingestion and Replication might truncate or re-create the target tables depending on the specific circumstances.
Schema	Select the target schema in which Database Ingestion and Replication creates the target tables.

The following table describes advanced target properties that appear under **Advanced**:

Property	Description
Add Last Replicated Time	Select this check box to add a metadata column that records the timestamp at which a record was inserted or last updated in the target table. For initial loads, all loaded records have the same timestamp. For incremental and combined initial and incremental loads, the column records the timestamp of the last DML operation that was applied to the target. By default, this check box is not selected.
Prefix for Metadata Columns	Add a prefix to the names of the added metadata columns to easily identify them and to prevent conflicts with the names of existing columns. Do not include special characters in the prefix. Otherwise, task deployment will fail. The default value is INFA_.

Oracle target properties

When you define a database ingestion and replication task, you must enter some properties for your Oracle target on the **Target** page of the task wizard. The properties vary slightly by load type.

The following table describes the Oracle target properties that appear under **Target**:

Property	Description
Target Creation	The only available option is **Create Target Tables**, which generates the target tables based on the source tables. **Note:** After the target table is created, Database Ingestion and Replication intelligently handles the target tables on subsequent job runs. Database Ingestion and Replication might truncate or re-create the target tables depending on the specific circumstances.
Schema	Select the target schema in which Database Ingestion and Replication creates the target tables.
Apply Mode	For incremental load and combined initial and incremental load jobs, indicates how source DML changes, including inserts, updates, and deletes, are applied to the target. Options are: - **Standard**. Accumulate the changes in a single apply cycle and intelligently merge them into fewer SQL statements before applying them to the target. For example, if an update followed by a delete occurs on the source row, no row is applied to the target. If multiple updates occur on the same column or field, only the last update is applied to the target. If multiple updates occur on different columns or fields, the updates are merged into a single update record before being applied to the target. - **Audit**. Apply an audit trail of every DML operation made on the source tables to the target. A row for each DML change on a source table is written to the generated target table along with the audit columns you select under the **Advanced** section. The audit columns contain metadata about the change, such as the DML operation type, time, owner, transaction ID, and generated ascending sequence number. Consider using Audit apply mode when you want to use the audit history to perform downstream computations or processing on the data before writing it to the target database or when you want to examine metadata about the captured changes. The default value is **Standard**. **Note:** This field does not appear if you selected **Query-based** as the CDC method on the **Source** page of the task wizard.

The following table describes the advanced target properties that you can set under **Advanced** if you set **Apply Mode** to **Audit**:

Field	Description
Add Operation Type	Select this check box to add a metadata column that records the source SQL operation type in the output that the job propagates to the target database or inserts into the target table. The job writes "I" for insert, "U" for update, or "D" for delete. By default, this check box is selected.
Add Operation Time	Select this check box to add a metadata column that records the source SQL operation timestamp in the output that the job propagates to the target table. By default, this check box is not selected.
Add Operation Owner	Select this check box to add a metadata column that records the owner of the source SQL operation in the output that the job propagates to the target. By default, this check box is not selected. This property is not available for jobs that have a MongoDB or PostgreSQL source. **Note:** This property is not supported for jobs that have a SQL Server source and use the CDC Tables capture method.

Field	Description
Add Operation Transaction Id	Select this check box to add a metadata column that includes the source transaction ID in the output that the job propagates to the target for SQL operations. By default, this check box is not selected.
Add Operation Sequence	Select this check box to add a metadata column that records a generated, ascending sequence number for each change operation that the job inserts into the target tables. The sequence number reflects the change stream position of the operation. By default, this check box is not selected.
Add Before Images	Select this check box to add _OLD columns with UNDO "before image" data in the output that the job inserts into the target tables. You can then compare the old and current values for each data column. For a delete operation, the current value will be null. By default, this check box is not selected.
Prefix for Metadata Columns	Add a prefix to the names of the added metadata columns to easily identify them and to prevent conflicts with the names of existing columns. The default value is INFA_.

Oracle Cloud Object Storage target properties

When you define a database ingestion and replication task that has an Oracle Cloud Object Storage target, you must enter some target properties on the **Target** tab of the task wizard.

Under **Target**, you can enter the following Oracle Cloud Object Storage target properties:

Property	Description
Output Format	Select the format of the output file. Options are: - **CSV** - **AVRO** - **PARQUET** The default value is **CSV**. **Note:** Output files in CSV format use double-quotation marks ("") as the delimiter for each field.
Add Headers to CSV File	If **CSV** is selected as the output format, select this check box to add a header with source column names to the output CSV file.
Avro Format	If you selected **AVRO** as the output format, select the format of the Avro schema that will be created for each source table. Options are: - **Avro-Flat**. This Avro schema format lists all Avro fields in one record. - **Avro-Generic**. This Avro schema format lists all columns from a source table in a single array of Avro fields. - **Avro-Nested**. This Avro schema format organizes each type of information in a separate record. The default value is **Avro-Flat**.
Avro Serialization Format	If **AVRO** is selected as the output format, select the serialization format of the Avro output file. Options are: - **None** - **Binary** - **JSON** The default value is **Binary**.

Property	Description
Avro Schema Directory	If **AVRO** is selected as the output format, specify the local directory where Database Ingestion and Replication stores Avro schema definitions for each source table. Schema definition files have the following naming pattern: *schemaname_tablename*.txt **Note:** If this directory is not specified, no Avro schema definition file is produced.
File Compression Type	Select a file compression type for output files in CSV or AVRO output format. Options are: - **None** - **Deflate** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Avro Compression Type	If **AVRO** is selected as the output format, select an Avro compression type. Options are: - **None** - **Bzip2** - **Deflate** - **Snappy** The default value is **None**, which means no compression is used.
Parquet Compression Type	If the **PARQUET** output format is selected, you can select a compression type that is supported by Parquet. Options are: - **None** - **Gzip** - **Snappy** The default value is **None**, which means no compression is used.
Deflate Compression Level	If **Deflate** is selected in the **Avro Compression Type** field, specify a compression level from 0 to 9. The default value is 0.
Task Target Directory	For incremental load and combined initial and incremental load tasks, the root directory for the other directories that hold output data files, schema files, and CDC cycle contents and completed files. You can use it to specify a custom root directory for the task. If you enable the **Connection Directory as Parent** option, you can still optionally specify a task target directory to use with the parent directory specified in the connection properties. This field is required if the {TaskTargetDirectory} placeholder is specified in patterns for any of the following directory fields.
Add Directory Tags	For incremental load and combined initial and incremental load tasks, select this check box to add the "dt=" prefix to the names of apply cycle directories to be compatible with the naming convention for Hive partitioning. This check box is cleared by default.
Connection Directory as Parent	Select this check box to use the directory value that is specified in the target connection properties as the parent directory for the custom directory paths specified in the task target properties. For initial load tasks, the parent directory is used in the **Data Directory** and **Schema Directory**. For incremental load and combined initial and incremental load tasks, the parent directory is used in the **Data Directory**, **Schema Directory**, **Cycle Completion Directory**, and **Cycle Contents Directory**. This check box is selected by default. If you clear it, for initial loads, define the full path to the output files in the **Data Directory** field. For incremental loads, optionally specify a root directory for the task in the **Task Target Directory**.

Property	Description

Data Directory	*For initial load tasks*, define a directory structure for the directories where Database Ingestion and Replication stores output data files and optionally stores the schema. To define directory pattern, you can use the following types of entries:
	- The placeholders {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the initial load job starts to transfer data to the target.
	- Specific directory names.
	- The toUpper() and toLower() functions, which force the values for an associated (*placeholder*) to uppercase or lowercase.
	Note: Placeholder values are not case sensitive.
	Examples:
	`myDir1/{SchemaName}/{TableName}`
	`myDir1/myDir2/{SchemaName}/{YYYY}/{MM}/{TableName}_{Timestamp}`
	`myDir1/{toLower(SchemaName)}/{TableName}_{Timestamp}`
	The default directory pattern is `{TableName)_{Timestamp}`.
	For incremental load and combined initial and incremental load tasks, define a custom path to the subdirectory that contains the cdc-data data files. To define the directory pattern, you can use the following types of entries:
	- The placeholders {TaskTargetDirectory}, {SchemaName}, {TableName), {Timestamp}, {YY}, {YYYY}, {MM}, and {DD}, where {YY}, {YYYY}, {MM}, and {DD} are for date elements. The {Timestamp} values are in the format yyyymmdd_hhmissms. The generated dates and times in the directory paths indicate when the CDC cycle started.
	If you include the toUpper or toLower function, put the placeholder name in parentheses and enclose the both the function and placeholder in curly brackets, as shown in the preceding example.
	- Specific directory names.
	The default directory pattern is `{TaskTargetDirectory}/data/{TableName}/data`
	Note: For Amazon S3, Flat File, Microsoft Azure Data Lake Storage Gen2, and Oracle Cloud Object Store targets, Database Ingestion and Replication uses the directory specified in the target connection properties as the root for the data directory path when **Connection Directory as Parent** is selected. For Google Cloud Storage targets, Database Ingestion and Replication uses the **Bucket** name that you specify in the target properties for the ingestion task. For Microsoft Fabric OneLake targets, the parent directory is the path specified in the **Lakehouse Path** field in the Microsoft Fabric OneLake connection properties.

Schema Directory	Specify a custom directory in which to store the schema file if you want to store it in a directory other than the default directory. For initial loads, previously used values if available are shown in a drop-down list for your convenience. This field is optional. For initial loads, the schema is stored in the data directory by default. For incremental loads and combined initial and incremental loads, the default directory for the schema file is `{TaskTargetDirectory}/data/{TableName}/schema` You can use the same placeholders as for the **Data Directory** field. Ensure that you enclose placeholders with curly brackets { }. If you include the toUpper or toLower function, put the placeholder name in parentheses and enclose the both the function and placeholder in curly brackets, for example: `{toLower(SchemaName)}` **Note:** Schema is written only to output data files in CSV format. Data files in Parquet and Avro formats contain their own embedded schema.
Cycle Completion Directory	For incremental load and combined initial and incremental load tasks, the path to the directory that contains the cycle completed file. Default is `{TaskTargetDirectory}/cycle/completed`.

Property	Description
Cycle Contents Directory	For incremental load and combined initial and incremental load tasks, the path to the directory that contains the cycle contents files. Default is `{TaskTargetDirectory}/cycle/contents`.
Use Cycle Partitioning for Data Directory	For incremental load and combined initial and incremental load tasks, causes a timestamp subdirectory to be created for each CDC cycle, under each data directory. If this option is not selected, individual data files are written to the same directory without a timestamp, unless you define an alternative directory structure.
Use Cycle Partitioning for Summary Directories	For incremental load and combined initial and incremental load tasks, causes a timestamp subdirectory to be created for each CDC cycle, under the summary contents and completed subdirectories.
List Individual Files in Contents	For incremental load and combined initial and incremental load tasks, lists individual data files under the contents subdirectory. If **Use Cycle Partitioning for Summary Directories** is cleared, this option is selected by default. All of the individual files are listed in the contents subdirectory unless you can configure custom subdirectories by using the placeholders, such as for timestamp or date. If **Use Cycle Partitioning for Data Directory** is selected, you can still optionally select this check box to list individual files and group them by CDC cycle.

Under **Advanced**, you can enter the following advanced target properties to add metadata columns for each delete operation or each DML change recorded in the audit table.

Field	Description
Add Operation Type	Select this check box to add a metadata column that records the source SQL operation type in the output that the job propagates to the target. For incremental loads, the job writes "I" for insert, "U" for update, or "D" for delete. For initial loads, the job always writes "I" for insert. By default, this check box is selected for incremental load and initial and incremental load jobs, and cleared for initial load jobs.
Add Operation Time	Select this check box to add a metadata column that records the source SQL operation timestamp in the output that the job propagates to the target. For initial loads, the job always writes the current date and time. By default, this check box is not selected.
Add Operation Owner	Select this check box to add a metadata column that records the owner of the source SQL operation in the output that the job propagates to the target. For initial loads, the job always writes "INFA" as the owner. By default, this check box is not selected. This property is not available for jobs that have a MongoDB or PostgreSQL source. **Note:** This property is not supported for jobs that have a SQL Server source and use the CDC Tables capture method.

Field	Description
Add Operation Transaction Id	Select this check box to add a metadata column that includes the source transaction ID in the output that the job propagates to the target for SQL operations. For initial loads, the job always writes "1" as the ID. By default, this check box is not selected.
Add Before Images	Select this check box to include UNDO data in the output that a job writes to the target. For initial loads, the job writes nulls. By default, this check box is not selected.

PostgreSQL target properties

When you define a database ingestion and replication task, you must enter some properties for your PostgreSQL target on the **Target** page of the task wizard.

The following table describes the target properties that appear under **Target**:

Property	Description
Target Creation	The only available option is **Create Target Tables**, which generates the target tables based on the source tables. **Note:** After the target table is created, Database Ingestion and Replication intelligently handles the target tables on subsequent job runs. Database Ingestion and Replication might truncate or re-create the target tables depending on the specific circumstances.
Schema	Select the target schema in which Database Ingestion and Replication creates the target tables.

Snowflake Data Cloud target properties

When you define a database ingestion and replication task, you must enter some properties for your Snowflake Data Cloud target on the **Target** page of the task wizard. The properties vary slightly by load type.

The following table describes the Snowflake target properties that appear under **Target**:

Property	Description
Target Creation	The only available option is **Create Target Tables**, which generates the target tables based on the source tables. **Note:** After the target table is created, Database Ingestion and Replication intelligently handles the target tables on subsequent job runs. Database Ingestion and Replication might truncate or re-create the target tables depending on the specific circumstances.
Schema	Select the target schema in which Database Ingestion and Replication creates the target tables.

Property	Description
Stage	The name of internal staging area that holds the data read from the source before the data is written to the target tables. This name must not include spaces. If the staging area does not exist, it will be automatically created.
Apply Mode	For incremental load and combined initial and incremental load jobs, indicates how source DML changes, including inserts, updates, and deletes, are applied to the target. Options are: - **Standard**. Accumulate the changes in a single apply cycle and intelligently merge them into fewer SQL statements before applying them to the target. For example, if an update followed by a delete occurs on the source row, no row is applied to the target. If multiple updates occur on the same column or field, only the last update is applied to the target. If multiple updates occur on different columns or fields, the updates are merged into a single update record before being applied to the target. - **Soft Deletes**. Apply source delete operations to the target as soft deletes. A soft delete marks the deleted row as deleted without actually removing it from the database. For example, a delete on the source results in a change record on the target with "D" displayed in the INFA_OPERATION_TYPE column. Consider using soft deletes if you have a long-running business process that needs the soft-deleted data to finish processing, to restore data after an accidental delete operation, or to track deleted values for audit purposes. **Note:** If you use **Soft Deletes** mode, you must not perform an update on the primary key in a source table. Otherwise, data corruption can occur on the target. - **Audit**. Apply an audit trail of every DML operation made on the source tables to the target. A row for each DML change on a source table is written to the generated target table along with the audit columns you select under the **Advanced** section. The audit columns contain metadata about the change, such as the DML operation type, time, owner, transaction ID, generated ascending sequence number, and before image. Consider using Audit apply mode when you want to use the audit history to perform downstream computations or processing on the data before writing it to the target database or when you want to examine metadata about the captured changes. The default value is **Standard**. **Note:** This field does not appear if you selected **Query-based** as the CDC method on the **Source** page of the task wizard.

The following table describes advanced target properties that appear under **Advanced**:

Property	Description
Add Last Replicated Time	Select this check box to add a metadata column that records the timestamp at which a record was inserted or last updated in the target table. For initial loads, all loaded records have the same timestamp, except for Snowflake targets that use the Superpipe option where minutes and seconds might vary slightly. For incremental and combined initial and incremental loads, the column records the timestamp of the last DML operation that was applied to the target. By default, this check box is not selected.
Add Operation Type	Select this check box to add a metadata column that records the source SQL operation type in the output that the job propagates to the target database or inserts into the audit table on the target system. This field is available only when the **Apply Mode** option is set to **Audit** or **Soft Deletes**. In Audit mode, the job writes "I" for inserts, "U" for updates, "E" for upserts, or "D" for deletes to this metadata column. In Soft Deletes mode, the job writes "D" for deletes or NULL for inserts and updates. When the operation type is NULL, the other "Add Operation..." metadata columns are also NULL. Only when the operation type is "D" will the other metadata columns contain non-null values. By default, this check box is selected. You cannot deselect it if you are using soft deletes.
Add Operation Time	Select this check box to add a metadata column that records the source SQL operation timestamp in the output that the job propagates to the target database or inserts into the audit table on the target system. This field is available only when **Apply Mode** is set to **Audit** or **Soft Deletes**. By default, this check box is not selected.
Add Operation Owner	Select this check box to add a metadata column that records the owner of the source SQL operation in the output that the job propagates to the target database or inserts into the audit table on the target system. This field is available only when **Apply Mode** is set to **Audit** or **Soft Deletes**. By default, this check box is not selected. This property is not available for jobs that have a MongoDB or PostgreSQL source. **Note:** This property is not supported for jobs that have a SQL Server source and use the CDC Tables capture method.
Add Operation Transaction Id	Select this check box to add a metadata column that includes the source transaction ID in the output that the job propagates to the target for SQL operations. This field is available only when **Apply Mode** is set to **Audit** or **Soft Deletes**. By default, this check box is not selected.
Add Operation Sequence	Select this check box to add a metadata column that records a generated, ascending sequence number for each change operation that the job inserts into the audit table on the target system. The sequence number reflects the change stream position of the operation. This field is available only when **Apply Mode** is set to **Audit**. By default, this check box is not selected.
Add Before Images	Select this check box to add _OLD columns with UNDO "before image" data in the output that the job inserts into the target tables. You can then compare the old and current values for each data column. For a delete operation, the current value will be null. This field is available only when **Apply Mode** is set to **Audit**. By default, this check box is not selected.

Property	Description
Prefix for Metadata Columns	Add a prefix to the names of the added metadata columns to easily identify them and to prevent conflicts with the names of existing columns. The default value is INFA_.
Superpipe	Select this check box to use the Snowpipe Streaming API to quickly stream rows of data directly to Snowflake Data Cloud target tables with low latency instead of first writing the data to stage files. This option is available for all load types. When you configure the target connection, select KeyPair authentication. By default, this check box is selected. Deselect it if you want to write data to intermediate stage files.
Merge Frequency	When **Superpipe** is selected, you can optionally set the frequency, in seconds, at which change data rows are merged and applied to the Snowflake target tables. This field applies to incremental load and combined initial and incremental load tasks. Valid values are 300 through 604800. Default is 3600 seconds.
Enable Case Transformation	By default, target table names and column names are generated in the same case as the corresponding source names, unless cluster-level or session-level properties on the target override this case-sensitive behavior. If you want to control the case of letters in the target names, select this check box. Then select a **Case Transformation Strategy** option. **Note:** This check box is not available if you selected the **Superpipe** option. You cannot enable case transformation if you are using the Superpipe option for Snowflake.
Case Transformation Strategy	If you selected **Enable Case Transformation**, select one of the following options to specify how to handle the case of letters in generated target table (or object) names and column (or field) names: - **Same as source**. Use the same case as the source table (or object) names and column (or field) names. - **UPPERCASE**. Use all uppercase. - **lowercase**. Use all lowercase. The default value is **Same as source**. **Note:** The selected strategy will override any cluster-level or session-level properties on the target for controlling case.

Configuring schedule and runtime options

On the **Schedule and Runtime Options** page of the database ingestion and replication task wizard, you can specify a schedule for running initial load jobs periodically and configure runtime options for jobs of any load type.

1. Under **Advanced**, optionally edit the **Number of Rows in Output File** value to specify the maximum number of rows that the database ingestion and replication task writes to an output data file.

   Note: Advanced options are not displayed for jobs that have an Apache Kafka target.

   For incremental load operations and combined initial and incremental load operations, change data is flushed to the target either when this number of rows is reached or when the flush latency period expires and the job is *not* in the middle of processing a transaction. The flush latency period is the time that the job waits for more change data before flushing data to the target. The latency period is internally set to 10 seconds and cannot be changed.

   Valid values are 1 through 100000000. The default value for Amazon S3, Microsoft Azure Data Lake Storage Gen2, and Oracle Cloud Infrastructure (OCI) Object Storage targets is 1000 rows. For the other targets, the default value is 100000 rows.

   Note: For Microsoft Azure Synapse Analytics targets, the data is first sent to a Microsoft Azure Data Lake Storage staging file before being written to the target tables. After data is written to the target, the entire contents of the table-specific directory that includes the staging files are emptied. For Snowflake targets, the data is first stored in an internal stage area before being written to the target tables.
2. For initial load jobs only, optionally clear the **File Extension Based on File Type** check box if you want the output data files for Flat File, Amazon S3, Google Cloud Storage, Microsoft Azure Data Lake Storage, Microsoft Fabric OneLake, or Oracle Cloud Object Storage targets to have the .dat extension. This check box is selected by default, which causes the output files to have file-name extensions based on their file types.

   Note: For incremental load jobs with these target types, this option is not available. Database Ingestion and Replication always uses output file-name extensions based on file type.

3. For database ingestion and replication incremental load tasks that have Amazon S3, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, Microsoft Fabric OneLake, or Oracle Cloud Object Storage targets, configure the following apply cycle options:

Option	Description
Apply Cycle Interval	Specifies the amount of time that must elapse before a database ingestion and replication job ends an apply cycle. You can specify days, hours, minutes, and seconds or specify values for a subset of these time fields leaving the other fields blank. The default value is 15 minutes.
Apply Cycle Change Limit	Specifies the total number of records in all tables of a database ingestion and replication job that must be processed before the job ends an apply cycle. When this record limit is reached, the database ingestion and replication job ends the apply cycle and writes the change data to the target. The default value is 10000 records. **Note:** During startup, jobs might reach this limit more frequently than the apply cycle interval if they need to catch up on processing a backlog of older data.
Low Activity Flush Interval	Specifies the amount of time, in hours, minutes, or both, that must elapse during a period of no change activity on the source before a database ingestion and replication job ends an apply cycle. When this time limit is reached, the database ingestion and replication job ends the apply cycle and writes the change data to the target. If you do not specify a value for this option, a database ingestion and replication job ends apply cycles only after either the **Apply Cycle Change Limit** or **Apply Cycle Interval** limit is reached. No default value is provided.

Notes:

- Either the **Apply Cycle Interval** or **Apply Cycle Change Limit** field must have a non-zero value or use the default value.
- An apply cycle ends when the job reaches any of the three limits, whichever limit is met first.

4. Under **Schema Drift Options**, if the detection of schema drift is supported for your source and target combination, specify the schema drift option to use for each of the supported types of DDL operations.

   Schema drift options are supported for the following source - target combinations and load types:

Source	Load Type	Target
Db2 for i	Incremental Combined initial and incremental	Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka (incremental loads only), Microsoft Azure Data Lake Storage, Microsoft Azure Synapse Analytics, Microsoft Fabric OneLake, Oracle, Oracle Cloud Object Storage, PostgreSQL, Snowflake, and SQL Server
Db2 for LUW	Incremental Combined initial and incremental	Snowflake
Db2 for z/OS, except Db2 11	Incremental Combined initial and incremental	Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka (incremental loads only), Microsoft Azure Data Lake Storage, Microsoft Azure Synapse Analytics, Microsoft Fabric OneLake, Oracle, Oracle Cloud Object Storage, Snowflake, and SQL Server

Source	Load Type	Target
Microsoft SQL Server	Incremental Combined initial and incremental	Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka (incremental loads only), Microsoft Azure Data Lake Storage, Microsoft Azure Synapse Analytics, Microsoft Fabric OneLake, Oracle, Oracle Cloud Object Storage, Snowflake, and SQL Server
Oracle	Incremental Combined initial and incremental	Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka (incremental loads only), Microsoft Azure Data Lake Storage, Microsoft Azure Synapse Analytics, Microsoft Fabric OneLake, Oracle, Oracle Cloud Object Storage, PostgreSQL, Snowflake, and SQL Server
PostgreSQL	Incremental Combined initial and incremental	Incremental loads: Amazon Redshift, Amazon S3, Databricks, Google BigQuery, Google Cloud Storage, Kafka (incremental loads only), Microsoft Azure Data Lake Storage, Microsoft Azure Synapse Analytics, Microsoft Fabric OneLake, Oracle, Oracle Cloud Object Storage, and Snowflake Combined initial and incremental loads: Oracle and Snowflake

The types of supported DDL operations are:

- Add Column
- Modify Column
- Drop Column
- Rename Column

Note: The Modify Column and Rename Column options are not supported and not displayed for database ingestion jobs that have Google BigQuery targets.

The following table describes the schema drift options that you can set for a DDL operation type:

Option	Description
Ignore	Do not replicate DDL changes that occur on the source database to the target. For Amazon Redshift, Kafka, Microsoft Azure Synapse Analytics, PostgreSQL, Snowflake and SQL Server targets, this option is the default option for the Drop Column and Rename Column operation types. For Amazon S3, Google Cloud Storage, Microsoft Azure Data Lake Storage, and Oracle Cloud Object Storage targets that use the CSV output format, the **Ignore** option is disabled. For the AVRO output format, this option is enabled.
Replicate	Replicate the DDL operation to the target. For Amazon S3, Google Cloud Storage, Microsoft Azure Data Lake Storage, Microsoft Fabric OneLake, and Oracle Cloud Object Storage targets, this option is the default option for all operation types. For other targets, this option is the default option for the Add Column and Modify Column operation types. **Restrictions:** - If you try to replicate a type of schema change that is not supported on the target, database ingestion and replication jobs associated with the task will end with an error. For example, if you select **Replicate** for Rename Column operations on Microsoft Azure Synapse Analytics targets, the jobs will end. - Add Column operations that add a primary-key column are not supported and can cause unpredictable results. - For Databricks targets, the **Replicate** option is not available for **Drop Column**. If you specify **Replicate** for the **Rename Column** option, you must specify the following Databricks table properties after the task is deployed and before the job runs: 'delta.minReaderVersion' = '2', 'delta.minReaderVersion' = '2', 'delta.minWriterVersion' = '5', and 'delta.columnMapping.mode' = 'name'. - Modify Column operations that change the NULL or NOT NULL constraint for a column are not replicated to the target by design because changing the nullibility of a target column can cause problems when subsequent changes are applied. - DDL operations are replicated to the target only after some data changes have been captured.
Stop Job	Stop the entire database ingestion job.
Stop Table	Stop processing the source table on which the DDL change occurred. When one or more of the tables are excluded from replication because of the **Stop Table** schema drift option, the job state changes to **Running with Warning**. **Important:** The database ingestion and replication job cannot retrieve the data changes that occurred on the source table after the job stopped processing it. Consequently, data loss might occur on the target. To avoid data loss, you will need to resynchronize the source and target objects that the job stopped processing. Use the **Resume With Options** > **Resync** option. For more information, see "Overriding schema drift options when resuming a database ingestion and replication job" on page 167.

5. For incremental load jobs that have an Apache Kafka target, configure the following checkpointing options:

Option	Description
Checkpoint All Rows	Indicates whether a database ingestion and replication job performs checkpoint processing for every message that is sent to the Kafka target. **Note:** If this check box is selected, the **Checkpoint Every Commit**, **Checkpoint Row Count**, and **Checkpoint Frequency (secs)** options are ignored.
Checkpoint Every Commit	Indicates whether a database ingestion and replication job performs checkpoint processing for every commit that occurs on the source.
Checkpoint Row Count	Specifies the maximum number of messages that a database ingestion and replication job sends to the target before adding a checkpoint. If you set this option to 0, a database ingestion and replication job does not perform checkpoint processing based on the number of messages. If you set this option to 1, a database ingestion and replication jobs add a checkpoint for each message.
Checkpoint Frequency (secs)	Specifies the maximum number of seconds that must elapse before a database ingestion and replication job adds a checkpoint. If you set this option to 0, a database ingestion and replication job does not perform checkpoint processing based on elapsed time.

6. Under **Schedule**, if you want to run job instances for an initial load task based on an existing schedule instead of manually starting the job after it is deployed from one of the monitoring interfaces, select **Run this task based on a schedule** and then select a predefined schedule. The default option is **Do not run this task based on a schedule**.

   This field is unavailable for incremental load and combined initial and incremental load tasks.

   You can view and edit the schedule options in Administrator. If you edit the schedule, the changes will apply to all jobs that use the schedule. If you edit the schedule after deploying the task, you do not need to redeploy the task.

   If the schedule criteria for running the job is met but the previous job run is still active, Database Ingestion and Replication skips the new job run.

7. Under **Custom Properties**, you can specify custom properties that Informatica provides to meet your special requirements. To add a property, in the **Create Property** fields, enter the property name and value. Then click **Add Property**.

   Specify these properties only at the direction of Informatica Global Customer Support. Usually, these properties address unique environments or special processing needs. You can specify multiple properties, if necessary. A property name can contain only alphanumeric characters and the following special characters: periods (.), hyphens (-), and underscores (_).

   Tip: To delete a property, click the Delete icon button at the right end of the property row in the list.

8. Click **Save**.

Deploying a database ingestion and replication task

After you define a database ingestion and replication task and save it, deploy the task to create an executable job instance on the on-premises system that contains the Secure Agent and the Database Ingestion agent

service and DBMI packages. You must deploy the task before you can run the job. The deploy process also validates the task definition.

Before you deploy a task that has a Microsoft Azure Synapse Analytics or Snowflake target, drop any existing target tables that do not match the structure of the source tables, for example, because of added or dropped source columns or altered column null constraints or data types. When you deploy the task, the target tables are generated based the latest source structure.

▶ To deploy a database ingestion and replication task, in the task wizard, save the completed task definition and then click **Deploy**.

After you deploy a task successfully, the associated job instance is in the Deployed state. You can run it from the **My Jobs** page that's accessed from the navigation bar of the Home page, from the **All Jobs** page in the Monitor service, or from the **All Jobs** tab on the **Data Ingestion and Replication** page in Operational Insights.

Deployment considerations:

- If you included spaces in the database ingestion and replication task name, the spaces are omitted from the corresponding job name for the generated job instance.
- If the deployment process fails, the job status switches to Failed. You can then Undeploy the job. To diagnose the error, download the error log from one of the monitoring interfaces: 1) the **My Jobs** page accessed from the unified Home page, 2) the **All Jobs** page in Monitor service, or 3) the **All Jobs** tab on the **Data Ingestion and Replication** page in Operational Insights. Click the Actions menu for the job and select **Error Log** or **Download Log**. After you resolve the problem, deploy the task again from the database ingestion and replication task wizard.
- If you undeploy a job and then want to run a job for the associated database ingestion and replication task again, you must deploy the task again to create a new job instance. The new job instance name ends with an incremented number in the format *taskname-job_instance_number*. The job instance number is incremented each time you deploy the task by adding 1 to the maximum instance number across all database ingestion and replication jobs.
- If the Secure Agent is restarted while the task is deploying, the job status switches to Failed. Avoid restarting the Secure Agent while tasks are being deployed.
- If a task appears to be hung in the Deploying state, restart the Secure Agent. The associated job instance acquires the status of Failed. You can undeploy it and then deploy it again.

Running database ingestion and replication jobs

You can run a deployed database ingestion and replication job from one of the monitoring interfaces, or when you create a database ingestion and replication initial load task, you can specify a schedule for running the job instances associated with a task.

Running a deployed database ingestion and replication job

You can run a database ingestion and replication job that has been previously deployed and is in a state other than Undeployed.

You can issue the Run command from any of the following interfaces: the **My Jobs** page that's accessed from the navigation bar of the Home page, the **All Jobs** page in the Monitor service, or the **All Jobs** tab on the **Data Ingestion and Replication** page in Operational Insights

1. Navigate to the row for the job that you want to run.

2. In the Actions menu for the row, click **Run**.

   A subtask is started for each source table.

   Notes:

   - If the initial load portion of a combined initial and incremental load job fails to load data from a source table to a target table, the database ingestion and replication job retries the subtask for the table up to three times. The interval between retries is a minimum of 60 seconds. If all of the initial load retries fail, the subtask acquires the state of **Error** and the table is excluded from replication. The job then tries to proceed with incremental loading. In this case, the job status changes to **Running with Warning**.
   - If an initial load job detects inconsistencies between column definitions in the source and target tables, the job drops the target table and then re-creates it to be consistent with the source table, prior to loading source data to the target.
   - For initial load and combined tasks, the initial load might take a long time to perform if the source tables contain many rows.

Running initial load jobs based on a schedule

When you configure a database ingestion and replication initial load task, you can specify a schedule for running job instances associated with the task. You must have previously defined the schedule in Administrator.

1. In Administrator, create the schedule if it does not already exist.

   You can set schedule options to start a job instance on a specific date and time, plus 10 seconds, based on a specific time zone and to run job instances on a reoccurring basis.
2. When you configure a database ingestion and replication initial load task in Data Ingestion and Replication, on the **Schedule and Runtime Options** page, select **Run this task based on a schedule** and then select the schedule.

For more information, see "Scheduling" in Administrator help.

Managing database ingestion and replication jobs

After you configure and run database ingestion and replication tasks, you might occasionally need to perform some job management tasks such stopping, resuming, undeploying, or redeploying jobs.

Stopping a database ingestion and replication job

You can stop a database ingestion and replication job of any load type that is in the **Up and Running**, **Running with Warning**, or **On Hold** state.

For an incremental load job, the job stops after a checkpoint is taken. A checkpoint records the point in the change stream where incremental processing left off for recovery purposes.

For a combined initial and incremental load job, initial load subtasks that are running are allowed to run to completion and Initial load subtasks that are not running remain in their current states. For the incremental load portion of the job, a checkpoint is written to the checkpoint file or target recovery table before the job stops. The database ingestion and replication job will not be able to record a checkpoint unless a change record has been processed for at least one of the tables in the job during the first job run after deployment. If

a checkpoint is not available, the job resumes processing from the configured restart point, which is the latest available position in the change stream by default.

For an initial load job, any *running* subtasks are allowed to run to completion and then the job stops. Non-running subtasks remain in their current states.

1. Navigate to the row for the job that you want to stop in any of the following monitoring interfaces:
   - **My Jobs** page that's accessed from the navigation bar of the Home page
   - **All Jobs** page in the Monitor service
   - **All Jobs** tab on the **Data Ingestion and Replication** page in Operational Insights
2. Click the Actions menu for the job and select **Stop**.

   The job state switches to **Stopping** and then to **Stopped**.

   Tip: If the Stop operation is taking too long, you can abort the job.

Aborting a database ingestion and replication job

You can abort a database ingestion and replication job that is in the **Up and Running**, **On Hold**, **Running with Warning**, or **Stopping** state.

For an incremental load job, the job stops immediately after a checkpoint is taken. A checkpoint records the point in the change stream where incremental processing left off for recovery purposes.

For a combined initial and incremental load job, any *running* initial load subtasks stop immediately. For the incremental load portion of the job, a checkpoint is taken and then the job stops.

For an initial load job, any *running* subtasks stop immediately and then the job stops. Non-running subtasks remain in their current states.

1. Navigate to the row for the job that you want to abort in any of the following monitoring interfaces:
   - **My Jobs** page that's accessed from the navigation bar of the Home page
   - **All Jobs** page in the Monitor service
   - **All Jobs** tab on the **Data Ingestion and Replication** page in Operational Insights
2. From the **Actions** menu for the job, select **Abort**.

   The job state switches to **Aborting** and then to **Aborted**.

   For initial load jobs, the state of started and running subtasks switches to **Aborted**. For incremental load or combined initial and incremental load jobs, the state of subtasks switches to **Stopped**.

Resuming a database ingestion and replication job

You can resume a database ingestion and replication job that is in the Stopped, Aborted, or Failed state.

When you resume an initial load job that has multiple subtasks, Database Ingestion and Replication starts only the subtasks that are in a Failed, Stopped, Aborted, or Queued state.

When you resume an incremental load job or a combined initial and incremental load job, Database Ingestion and Replication resumes replicating source data change from the last position recorded in the checkpoint file or target recovery table. A checkpoint will not be available unless a change record was processed for at least one of the tables during the first job run after deployment. If a checkpoint is not available, the job resumes processing from the configured restart point, which is the latest available position in the change stream by default.

Note: For initial load jobs, the **Run** command might also be available. Click **Run** if you want the database ingestion and replication job to truncate all of the target tables and then reload the source data to the target tables.

1. Navigate to the row for the job that you want to resume in any of the following monitoring interfaces:
   - **My Jobs** page that's accessed from the navigation bar of the Home page
   - **All Jobs** page in the Monitor service
   - **All Jobs** tab on the **Data Ingestion and Replication** page in Operational Insights
2. In the **Actions** menu for the row, click **Resume**.

   Note: The **Resume** command is not available if the job is in the **Failed** state because the task deployment failed.

   A subtask is started for each source table.

   If an error occurs, an error message is displayed at the top of the page.

Overriding schema drift options when resuming a database ingestion and replication job

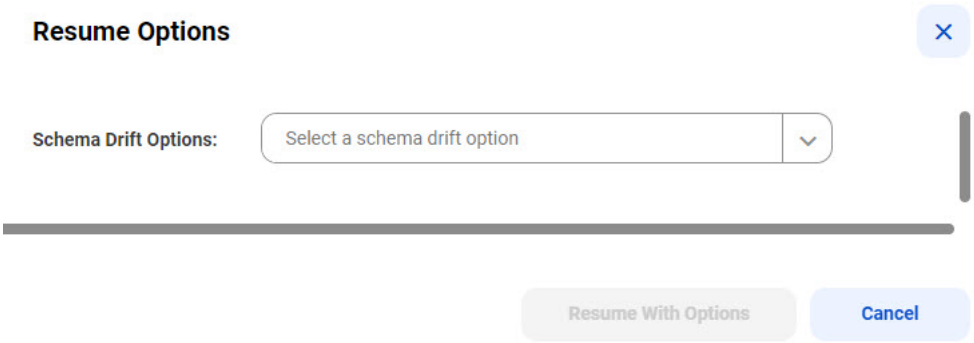
You can override the schema drift options when you resume a database ingestion and replication job that is in the Stopped, Aborted, or Failed state. The overrides affect only those tables that are currently in the Error state because of the **Stop Table** or **Stop Job** Schema Drift option. Use the overrides to correct or resolve these errors.

You can override schema drift options and resume an incremental load job or a combined initial and incremental load job from the **All Jobs** tab on the **Data Ingestion and Replication** page in Operational Insights.

1. Navigate to the row for the job that you want to resume with an override.
2. Click the Actions menu for the row and select **Resume With Options**.

   Note: The **Resume With Options** command is not available if the job is in the **Failed** state because the task deployment failed.

   The **Resume Options** dialog box appears.



3. In the **Schema Drift Options** list, select the schema drift option that will be used to process the DDL operation on the source that caused the database ingestion and replication job to stop.

The following table describes the schema drift options:

Option	Description
Ignore	Do not replicate DDL changes that occur on the source database to the target.
Stop Table	Stop processing the source table on which the DDL change occurred. **Important:** The database ingestion and replication job cannot retrieve the data changes that occurred on the source table after the job stopped processing it. Consequently, data loss might occur on the target. To avoid data loss, you will need to resynchronize the source and target objects that the job stopped processing. Use the **Resume With Options** > **Resync** option.
Resync	Resynchronize the target tables with the latest source table definitions, including any DDL changes that the schema drift ignored. Use this option for tables that the job stopped processing because of the **Stop Table** setting for a **Schema Drift** option. **Important:** This option is available only for combined initial and incremental load jobs.
Resync (refresh)	For database ingestion and replication combined load jobs that have an Oracle or SQL Server source, use this option to resynchronize the target tables with the latest source table definitions, including any DDL changes that schema drift ignored. After the target tables are refreshed, the structure of source and target tables match. This option mimics the behavior of the **Resync** option.
Resync (retain)	For database ingestion and replication combined load jobs that have an Oracle or SQL Server source, use this option to resynchronize the same columns that have been processed for CDC, retaining the current structure of the source and target tables. No checks for changes to the source or target table definitions are performed. If source DDL changes affected the source table structure, those changes are not processed.
Replicate	Allow the database ingestion and replication job to replicate the DDL change to the target. **Important:** If you specify the **Replicate** option for Rename Column operations on Microsoft Azure Synapse Analytics targets, the job will end with an error.

4. Click **Resume With Options**.

   The resumed job will use the schema drift option that you specified in step 3 to process the schema change that caused the job to stop. Thereafter, the schema drift options that you specified when creating the task take effect again.

   Important: Database Ingestion and Replication processes a schema change to a source table only after a DML operation occurs on the table. Therefore, after you resume a job, the table subtask state remains unchanged until the first DML operation occurs on the table.

Redeploying a database ingestion and replication job

Redeploy a database ingestion and replication job after editing available fields in the associated database ingestion and replication task so that the new settings can take effect.

You can edit some but not all of the fields in an ingestion task definition that has been previously deployed, without first undeploying the job. You can add a source table and change any of the runtime and target options that are available for editing. For example, you might want to reset some target options to test the effects of different settings.

The redeploy operation stops each job subtask for a source table, deploys the updated database ingestion and replication task, and automatically starts the subtasks that were stopped and any new subtasks for added source tables.

1. Navigate to the row for the job that you want to redeploy in any of the following monitoring interfaces:
   - **My Jobs** page that's accessed from the navigation bar of the Home page
   - **All Jobs** page in the Monitor service
   - **All Jobs** tab on the **Data Ingestion and Replication** page in Operational Insights
2. In the **Actions** menu for the row, select **Redeploy**.

   The job instance automatically starts running.
   If the job was running when you selected **Redeploy**, Database Ingestion and Replication stops the job and then redeploys the database ingestion and replication task and restarts the job.

Notes:

- For incremental load jobs and combined initial and incremental load jobs, the redeploy operation does not change the list of selected tables that was created during a previous deployment. To update the list of tables, edit the table selection rules in the associated task and then redeploy the job. You must make an update to the table selection rules, even if you added a table that matches the existing table selection rules.
- For a database ingestion and replication combined initial and incremental load job, if you select additional columns for a source table for which you've previously selected a subset of columns and then redeploy the job, the job triggers a resync operation to get the data for the added columns and write it to the target. If you deselect columns for a table, the resync operation is not triggered. Instead, the deselected columns are processed based on the schema drift **Drop Column** setting. For an initial load job, the job writes the content for added columns to the target the next time you run the job.
- If you redeploy an incremental load job after making some column DDL changes to tables for which you've previously selected a subset of columns, the job attempts to process the changes based on the schema drift options that you set on the **Schedule and Runtime Options** page of the task wizard. However, if the column selection and DDL changes are made at the same time, the results might be incorrect. If you redeploy a combined initial and incremental load task in the same situation, a resync operation is automatically triggered to make the source and target consistent.
- For jobs with Microsoft Azure Synapse Analytics or Snowflake Cloud Data Warehouse targets, the redeploy operation also validates that the target tables exist and creates new ones if table selection rules have changed.

Undeploying a database ingestion and replication job

Undeploy a database ingestion and replication job if you no longer need to run the job, the job is in the Failed state, or you need to change a connection or property in the associated task that cannot be edited without first undeploying the job.

Before you attempt to undeploy a job, ensure that it is not running.

After the job is undeployed, you cannot run it again or redeploy it. If you want to run a job for the associated database ingestion and replication task again, you must deploy the task again from the task wizard to create a new job instance. For example, if you want to change the target connection, undeploy the job, edit the task to change the connection, deploy the task again, and then run the new job instance.

1. Navigate to the row for the job that you want to undeploy in any of the following monitoring interfaces:
   - **My Jobs** page that's accessed from the navigation bar of the Home page

- **All Jobs** page in the Monitor service
- **All Jobs** tab on the **Data Ingestion and Replication** page in Operational Insights

2. In the **Actions** menu for the row, click **Undeploy**.

   If the undeploy operation fails, the job status switches to Failed, even if it was in the Aborted state, or remains as Failed.

   Note: After undeploying jobs, do not immediately shut down the Secure Agent. Database Ingestion and Replication requires some time to clean up files for tasks in the /root/infaagent/apps/Database_Ingestion/data/tasks directory.

Resynchronizing source and target objects

You can resynchronize source and target objects for a subtask that is part of a running database ingestion and replication combined initial and incremental load job. The subtask must be in a state other than Queued or Starting.

For example, you might want to resynchronize the target with the source if initial load or incremental load processing failed or if you want to start the job over again from a specific restart point.

Important: To resynchronize tables that stopped and are currently in the **Error** state because of the **Schema Drift** setting of **Stop Table**, you must use the **Resume With Options** > **Resync** option in the Actions menu. For more information, see "Overriding schema drift options when resuming a database ingestion and replication job" on page 167.

1. Drill down on the database ingestion and replication job that you want to resynchronize from one of the following monitoring interfaces:
   - **My Jobs** page that's accessed from the navigation bar of the Home page
   - **All Jobs** page in the Monitor service
   - **All Jobs** tab on the **Data Ingestion and Replication** page in Operational Insights

   The job must be in the **Up and Running** state and be for a combined initial and incremental load operation.
2. Click the **Object Detail** tab.
3. In the subtask row for the source and target objects that you want to resynchronize, click the **Actions** menu and select **Resync**. The resync operation resynchronizes the target table with the latest source table definition, including any DDL changes.

   Note: For the Actions menu and **Resync** option to be available, the subtask must be in a state other than Queued or Starting.

   If you are resynchronizing a subtask for a database ingestion and replication combined load job that has an Oracle or SQL Server source, use one of the following resync options instead of the **Resync** option:
   - **Resync (refresh)** - Use this option to resynchronize the target table with the latest source table definition, including any DDL changes that schema drift ignored. After the target table is refreshed, the target table structure matches the current source table structure. This option mimics the behavior of the **Resync** option.
   - **Resync (retain)** - Use this option to resynchronize the same columns that have been processed for CDC, retaining the current structure of the source and target tables. No checks for changes in the source or target table definitions are performed. If source DDL changes affected the source table structure, those changes are not processed.

Notes:

- If the source table contains many rows, the resynchronization might take a long time to perform.

- If the source table schema does not match the target table schema, the ingestion subtask drops the target table and creates a new table that matches the source schema. Regardless of whether the target tables are re-created, the subtask truncates the target tables and then reloads source data to the tables.
- When you resync a database ingestion and replication subtask with a Snowflake target and use Audit apply mode, you can retain the audit information. Data Ingestion and Replication re-creates the target table and renames the existing table that contains the audit information using an appended timestamp in the format <target_table_name>_<current_UTC_timestamp>. If you want the audit information in the new target table, you need to load it, for example, with a join operation. If adding the timestamp to the existing table name causes the name to exceed the maximum number of characters, the subtask fails with an error. If you enable schema drift and a schema drift change, such as Add Column, occurs, the new column will be in the re-created target table but not in the renamed table. To enable this behavior, set the backupTargetTableBeforeResync custom property to **true** on the **Target** page of the task wizard.

  Consider the following limitations when you resync a combined load job that has existing audit information:

  - Storing the existing table with the audit information on the target consumes extra database storage.
  - To obtain a unified view of audit information, you need to join the multiple versions of the target tables.

Restart and recovery for incremental change data processing

Database Ingestion and Replication can restart incremental load and combined initial and incremental load jobs that stopped because of an error or a user stop request without losing change data.

After the first job run, Database Ingestion and Replication continually records an identifier for the processing position in the change stream as changes are applied to the target. For file-based targets such as Amazon S3, Azure Data Lake Storage, Google Cloud Storage, Kafka, and Oracle Cloud Object Storage, the identifier is stored in a checkpoint file. For database targets, the identifier is stored in a generated recovery table, called INFORMATICA_CDC_RECOVERY, on the target.

Note: For the first run of an incremental load job, Database Ingestion and Replication uses the start point that you set in the **Initial Start Point for Incremental Load** field when defining the database ingestion and replication task.

If incremental change data processing ends abnormally or in response to a user stop or abort request and you then resume the job, the job resumes from the last position saved to the checkpoint file or recovery table. A checkpoint will not be available unless a change record was processed for at least one of the tables during the first job run after deployment. If a checkpoint is not available, the job resumes processing from the configured restart point, which is the latest available position in the change stream by default.

Running data validation for a database ingestion and replication jobs

For initial load jobs that completed successfully, you can run data validation to compare the source and target data. Data validation is available only for initial load jobs that have an Oracle or a SQL Server source and a Snowflake target.

Considerations:

- The availability of the data validation feature is controlled by an organization-level feature flag. If this functionality is not available for your organization but you want to use it, contact Informatica Global Customer Support.
- When you run data validation for Database Ingestion and Replication, you will be charged per the CPU consumption on the Data Validation service side.

- The source and target connections defined in the task for which you want to run the data validation must be on the same Secure Agent. You must enable the Data Validation service on the Secure Agent.
- The source and target schemas specified in the task definition must be the same as the schemas used in the source and target connection properties.
- In the Snowflake Data Cloud connection properties, enter the database and schema name in the **Additional JDBC URL Parameters** field in the following format:

  `db=<database_name>&schema=<schema_name>`
- For data validation to run successfully, the source table and column names cannot contain any special characters. Otherwise, data validation fails.
- To prevent false alarms that result from validating unsupported data types, you can exclude these data types by using the datavalidation.datatypes.skip custom property. On the **Schedule and Runtime Options** page of the task wizard, enter datavalidation.datatypes.skip as the property name and a comma-separated list of data types as the property value.

1. To display the job details, drill down on a job from the **My Jobs** page in the Data Integration service, the **All Jobs** page in the Monitor service, or from the **Data Ingestion and Replication** page in Operational Insights service.
2. On the **Object Detail** pane, navigate to the subtask row for which you want to run data validation. In the Actions menu for the row, select **Run Data Validation**.

   Note: For the **Run Data Validation** option to be available, the task must have the status of**Completed**.
3. Configure how the data should be validated:
   a. Select the Flat file connection.

      This connection will be used to store the data validation results.

      Note: The Flat file connection and the database ingestion and replication job must be on the same runtime environment.

   b. In the **Sample** field, select the option for sampling the size of the data for comparison. The default value is **Last 1000 Rows**.
4. Click **Run**.

   The data validation process starts. The **Data Validation** column in the **Object Detail** pane shows the data validation status for the selected task.
   If data validation processing completes successfully, you can click the **Success** status to view the Data Validation Summary. The summary contains the results of the row count validation and the cell-to-cell comparison.

   To download a detailed data validation report, click the Download icon. The report highlights any missing or modified rows and columns based on a comparison of the source and target tables.

   If an error occurred during the data validation processing, click the Download icon next to the **Error** status to view the error message.

Default Data Type Mappings

This reference provides default data-type mappings for relational sources and Amazon Redshift, Databricks, Google BigQuery, Microsoft Azure Synapse Analytics, Microsoft SQL Server, Oracle, PostgreSQL, and Snowflake targets. When Database Ingestion and Replication generates the target tables, it uses these mappings.

When you configure a target, you can optionally define data-type mapping rules to customize the default mappings of source data types to target data tables. For more information, see "Customizing data type mappings" on page 120.

If a source data type is not listed, Database Ingestion and Replication either cannot extract data from the source columns with this data type or cannot apply the extracted data to any appropriate target data type.

Db2 for i Source and Amazon Redshift Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for i source and an Amazon Redshift target:

Db2 for i Source Data Type	Amazon Redshift Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 32766	binary varying(*size*), 1 <= size <= 32766
char(*size*) for bit data, 1 <= size <= 32766	binary varying(*size*), 1 <= size <= 32766
char(*size*), 1 <= size <= 1024	character(*size*), 4 <= size <= 4096
char(*size*), 1025 <= size <= 32766	character varying(*size*), 4100 <= size <= 65535
date	date
decfloat(*precision*), 16 <= p <= 34	character varying(255)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
decimal(p,s), 38 <= p <= 63, 0 <= s <= 62	character varying(*size*), 40 <= size <= 65
float	double precision
integer	integer
long varbinary	binary varying(1024000)
long varchar	character varying(65535)
long varchar for bit data	binary varying(1024000)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
numeric(p,s), 38 <= p <= 63, 0 <= s <= 62	character varying(*size*), 40 <= size <= 65
real	real
rowid	binary varying(40)
smallint	smallint
time	time without time zone
timestamp(*precision*), 0 <= p <= 6	timestamp without time zone

Db2 for i Source Data Type	Amazon Redshift Target Data Type
timestamp(*precision*), 7 <= p <= 12	character varying(*size*), 27 <= size <= 32
varbinary(*size*), 1 <= size <= 32740	binary varying(*size*), 1 <= size <= 32740
varchar(*size*) for bit data, 1 <= size <= 32740	binary varying(*size*), 1 <= size <= 32740
varchar(*size*), 1 <= size <= 32740	character varying(*size*), 4 <= size <= 65535

Unsupported source data types
Database Ingestion and Replication does not support the following Db2 for i data types:

- BLOB
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for i Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for i source and a Databricks target:

Db2 for i Source Data Type	Databricks Target Data Type
bigint	long
binary(*size*), 1 <= size <= 32766	binary
char(*size*) for bit data, 1 <= size <= 32766	binary
char(*size*), 1 <= size <= 32766	string
date	string
decfloat(*precision*), 16 <= p <= 34	string
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
decimal(p,s), 39 <= p <= 63, 0 <= s <= 62	string
float	double
integer	integer

Db2 for i Source Data Type	Databricks Target Data Type
long varbinary	binary
long varchar	string
long varchar for bit data	binary
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 63, 0 <= s <= 62	string
real	float
rowid	binary
smallint	integer
time	string
timestamp(*precision*), 0 <= p <= 6	timestamp
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= s <= 32740	binary
varchar(*size*) for bit data, 1 <= size <= 32740	binary
varchar(*size*), 1 <= size <= 32740	string

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for i data types:

- BLOB
- CLOB
- DATALINK
- DECIMAL or NUMERIC with a scale greater than the precision
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for i Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for i source and a Google BigQuery target:

Db2 for i Source Data Type	Google BigQuery Target Data Type
bigint	int64
binary(*size*), 1 <= size <= 32766	bytes
char(*size*) for bit data, 1 <= size <= 32766	bytes
char(*size*), 1 <= size <= 32766	string
date	date
decfloat(*precision*), 16 <= p <= 34	string
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric

Db2 for i Source Data Type	Google BigQuery Target Data Type
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(5,s), 1 <= s <= 5	numeric
decimal(6,s), 1 <= s <= 6	numeric
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 48, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 49, 0 <= s <= 30	bignumeric
decimal(p,s), 31 <= p <= 50, 0 <= s <= 31	bignumeric
decimal(p,s), 32 <= p <= 51, 0 <= s <= 32	bignumeric
decimal(p,s), 33 <= p <= 52, 0 <= s <= 33	bignumeric
decimal(p,s), 34 <= p <= 53, 0 <= s <= 34	bignumeric
decimal(p,s), 35 <= p <= 54, 0 <= s <= 35	bignumeric
decimal(p,s), 36 <= p <= 55, 0 <= s <= 36	bignumeric
decimal(p,s), 37 <= p <= 56, 0 <= s <= 37	bignumeric
decimal(p,s), 38 <= p <= 56, 0 <= s <= 38	bignumeric
decimal(p,s), 39 <= p <= 40, 0 <= s <= 39	string
decimal(p,s), 39 <= p <= 40, 21 <= s <= 38	bignumeric
decimal(p,s), 40 <= p <= 41, 0 <= s <= 40	string
decimal(p,s), 41 <= p <= 42, 0 <= s <= 41	string
decimal(p,s), 42 <= p <= 43, 0 <= s <= 42	string
decimal(p,s), 43 <= p <= 44, 0 <= s <= 43	string
decimal(p,s), 44 <= p <= 63, 0 <= s <= 44	string

Db2 for i Source Data Type	Google BigQuery Target Data Type
decimal(p,s), 45 <= p <= 46, 0 <= s <= 45	string
decimal(p,s), 46 <= p <= 47, 0 <= s <= 46	string
decimal(p,s), 47 <= p <= 48, 0 <= s <= 47	string
decimal(p,s), 48 <= p <= 49, 0 <= s <= 48	string
decimal(p,s), 49 <= p <= 50, 0 <= s <= 49	string
decimal(p,s), 50 <= p <= 51, 0 <= s <= 50	string
decimal(p,s), 51 <= p <= 52, 0 <= s <= 51	string
decimal(p,s), 52 <= p <= 53, 0 <= s <= 52	string
decimal(p,s), 53 <= p <= 54, 0 <= s <= 53	string
decimal(p,s), 54 <= p <= 55, 0 <= s <= 54	string
decimal(p,s), 55 <= p <= 56, 0 <= s <= 55	string
decimal(p,s), 56 <= p <= 57, 0 <= s <= 56	string
decimal(p,s), 56 <= p <= 63, 39 <= s <= 62	string
decimal(p,s), 57 <= p <= 58, 0 <= s <= 57	string
decimal(p,s), 58 <= p <= 59, 0 <= s <= 58	string
decimal(p,s), 59 <= p <= 60, 0 <= s <= 59	string
decimal(p,s), 60 <= p <= 61, 0 <= s <= 60	string
decimal(p,s), 61 <= p <= 62, 0 <= s <= 61	string
decimal(p,s), 62 <= p <= 63, 0 <= s <= 62	string
decimal(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
float	float64
integer	int64
long varbinary	bytes
long varchar	string
long varchar for bit data	bytes
numeric(1,1)	numeric
numeric(10,10)	bignumeric

Db2 for i Source Data Type	Google BigQuery Target Data Type
numeric(11,s), 10 <= s <= 11	bignumeric
numeric(12,s), 10 <= s <= 12	bignumeric
numeric(13,s), 10 <= s <= 13	bignumeric
numeric(14,s), 10 <= s <= 14	bignumeric
numeric(15,s), 10 <= s <= 15	bignumeric
numeric(16,s), 10 <= s <= 16	bignumeric
numeric(17,s), 10 <= s <= 17	bignumeric
numeric(18,s), 10 <= s <= 18	bignumeric
numeric(19,s), 10 <= s <= 19	bignumeric
numeric(2,s), 1 <= s <= 2	numeric
numeric(20,s), 10 <= s <= 20	bignumeric
numeric(21,s), 10 <= s <= 21	bignumeric
numeric(22,s), 10 <= s <= 22	bignumeric
numeric(23,s), 10 <= s <= 23	bignumeric
numeric(24,s), 10 <= s <= 24	bignumeric
numeric(25,s), 10 <= s <= 25	bignumeric
numeric(26,s), 10 <= s <= 26	bignumeric
numeric(27,s), 10 <= s <= 27	bignumeric
numeric(28,s), 10 <= s <= 28	bignumeric
numeric(3,s), 1 <= s <= 3	numeric
numeric(4,s), 1 <= s <= 4	numeric
numeric(5,s), 1 <= s <= 5	numeric
numeric(6,s), 1 <= s <= 6	numeric
numeric(7,s), 1 <= s <= 7	numeric
numeric(8,s), 1 <= s <= 8	numeric
numeric(p,0), 1 <= p <= 18	int64
numeric(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric

Db2 for i Source Data Type	Google BigQuery Target Data Type
numeric(p,s), 29 <= p <= 48, 0 <= s <= 29	bignumeric
numeric(p,s), 30 <= p <= 49, 0 <= s <= 30	bignumeric
numeric(p,s), 31 <= p <= 50, 0 <= s <= 31	bignumeric
numeric(p,s), 32 <= p <= 51, 0 <= s <= 32	bignumeric
numeric(p,s), 33 <= p <= 52, 0 <= s <= 33	bignumeric
numeric(p,s), 34 <= p <= 53, 0 <= s <= 34	bignumeric
numeric(p,s), 35 <= p <= 54, 0 <= s <= 35	bignumeric
numeric(p,s), 36 <= p <= 55, 0 <= s <= 36	bignumeric
numeric(p,s), 37 <= p <= 56, 0 <= s <= 37	bignumeric
numeric(p,s), 38 <= p <= 56, 0 <= s <= 38	bignumeric
numeric(p,s), 39 <= p <= 40, 0 <= s <= 39	string
numeric(p,s), 39 <= p <= 40, 21 <= s <= 38	bignumeric
numeric(p,s), 40 <= p <= 41, 0 <= s <= 40	string
numeric(p,s), 41 <= p <= 42, 0 <= s <= 41	string
numeric(p,s), 42 <= p <= 43, 0 <= s <= 42	string
numeric(p,s), 43 <= p <= 44, 0 <= s <= 43	string
numeric(p,s), 44 <= p <= 63, 0 <= s <= 44	string
numeric(p,s), 45 <= p <= 46, 0 <= s <= 45	string
numeric(p,s), 46 <= p <= 47, 0 <= s <= 46	string
numeric(p,s), 47 <= p <= 48, 0 <= s <= 47	string
numeric(p,s), 48 <= p <= 49, 0 <= s <= 48	string
numeric(p,s), 49 <= p <= 50, 0 <= s <= 49	string
numeric(p,s), 50 <= p <= 51, 0 <= s <= 50	string
numeric(p,s), 51 <= p <= 52, 0 <= s <= 51	string
numeric(p,s), 52 <= p <= 53, 0 <= s <= 52	string
numeric(p,s), 53 <= p <= 54, 0 <= s <= 53	string
numeric(p,s), 54 <= p <= 55, 0 <= s <= 54	string

Db2 for i Source Data Type	Google BigQuery Target Data Type
numeric(p,s), 55 <= p <= 56, 0 <= s <= 55	string
numeric(p,s), 56 <= p <= 57, 0 <= s <= 56	string
numeric(p,s), 56 <= p <= 63, 39 <= s <= 62	string
numeric(p,s), 57 <= p <= 58, 0 <= s <= 57	string
numeric(p,s), 58 <= p <= 59, 0 <= s <= 58	string
numeric(p,s), 59 <= p <= 60, 0 <= s <= 59	string
numeric(p,s), 60 <= p <= 61, 0 <= s <= 60	string
numeric(p,s), 61 <= p <= 62, 0 <= s <= 61	string
numeric(p,s), 62 <= p <= 63, 0 <= s <= 62	string
numeric(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
real	float64
rowid	bytes
smallint	int64
time	time
timestamp(*precision*), 0 <= p <= 6	datetime
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= size <= 32740	bytes
varchar(*size*) for bit data, 1 <= size <= 32740	bytes
varchar(*size*), 1 <= size <= 32740	string

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for i data types:

- BLOB
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for i Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for i source and a Microsoft Azure Synapse Analytics target:

Db2 for i Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 8000	binary(*size*), 1 <= size <= 8000
binary(*size*), 8001 <= size <= 32766	varbinary(max)
char(*size*) for bit data, 1 <= size <= 8000	binary(*size*), 1 <= size <= 8000
char(*size*) for bit data, 8001 <= size <= 32766	varbinary(max)
char(*size*), 1 <= size <= 8000	char(*size*), 1 <= size <= 8000
char(*size*), 8001 <= size <= 32766	varchar(max)
date	date
decfloat(*precision*), 16 <= p <= 34	varchar(255)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
decimal(p,s), 39 <= p <= 63, 0 <= s <= 62	char(*size*), 40 <= size <= 65
float	float
integer	int
long varbinary	varbinary(max)
long varchar	varchar(max)
long varchar for bit data	varbinary(max)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 63, 0 <= s <= 62	char(*size*), 40 <= size <= 65
real	real
rowid	varbinary(40)
smallint	smallint
time	time(0)
timestamp(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
timestamp(*precision*), 8 <= p <= 12	char(*size*), 28 <= size <= 32
varbinary(*size*), 1 <= size <= 32740	varbinary(*size*), 1 <= size <= max

Db2 for i Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
varchar(*size*) for bit data, 1 <= size <= 32740	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 32740	varchar(*size*), 1 <= size <= max

Unsupported source data types
Database Ingestion and Replication does not support the following Db2 for i data types:

- BLOB
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for i Source and Microsoft SQL Server Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for i source and a Microsoft SQL Server target:

Db2 for i Source Data Type	Microsoft SQL Server Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 8000	varbinary(*n*), 1 <= n <= 8000
binary(*size*), 8001 <= size <= 32766	varbinary(max)
char(*size*) for bit data, 1 <= size <= 8000	varbinary(*n*), 1 <=n <= 8000
char(*size*) for bit data, 8001 <= size <= 32766	varbinary(max)
char(*size*), 1 <= size <= 8000	varchar(*n*), 1 <= n <= 8000
char(*size*), 8001 <= size <= 32766	varchar(max)
date	date
decfloat(*precision*), 16 <= p <= 34	varchar(255)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
decimal(p,s), 39 <= p <= 63, 0 <= s <= 62	char(*n*), 40 <= n <= 65
float	float

Db2 for i Source Data Type	Microsoft SQL Server Target Data Type
integer	int
long varbinary	varbinary(max)
long varchar	varchar(max)
long varchar for bit data	varbinary(max)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 63, 0 <= s <= 62	char(*n*), 40 <= n <= 65
real	real
rowid	varbinary(40)
smallint	smallint
time	time(0)
timestamp(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
timestamp(*precision*), 8 <= p <= 12	char(*n*), 28 <= n <= 32
varbinary(*size*), 1 <= size <= 32740	varbinary(*n*), 1 <= n <= max
varchar(*size*) for bit data, 1 <= size <= 32740	varbinary(*n*), 1 <= n <= max
varchar(*size*), 1 <= size <= 32740	varchar(*n*), 1 <= n <= max

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for i data types:

- BLOB
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for i Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for i source and an Oracle target:

Db2 for i Source Data Type	Oracle Target Data Type
bigint	number(19)
binary(*size*), 1 <= s <= 2000	raw(*size*), 1 <= size <= 2000
binary(*size*), 2001 <= s <= 32766	blob
char(*size*) for bit data, 1 <= s <= 32766	blob
char(*size*), 1 <= s <= 2000	char(s byte), 1 <= s <= 2000
char(*size*), 2001 <= s <= 4000	varchar2(s byte), 2001 <= s <= 4000
char(*size*), 4001 <= s <= 32766	clob
date	date
decfloat(*precision*), 16 <= p <= 34	char(255 char)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
decimal(p,s), 39 <= p <= 63, 0 <= s <= 62	char(s char), 40 <= s <= 65
float	binary_double
integer	number(10)
long varbinary	blob
long varchar	clob
long varchar for bit data	blob
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 63, 0 <= s <= 62	char(s char), 40 <= s <= 65
real	binary_double
rowid	blob
smallint	number(5)
time	char(8 char)
timestamp(0)	date
timestamp(*precision*), 1 <= p <= 9	timestamp(*precision*), 1 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(s char), 30 <= s <= 32

Db2 for i Source Data Type	Oracle Target Data Type
varbinary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
varbinary(*size*), 2001 <= size <= 32740	blob
varchar(*size*) for bit data, 1 <= size <= 32740	blob
varchar(*size*), 1 <= size <= 4000	varchar2(s byte), 1 <= s <= 4000
varchar(*size*), 4001 <= size <= 32740	clob

Unsupported source data types
Database Ingestion and Replication does not support the following Db2 for i data types:

- BLOB
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for i Source and PostgreSQL Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for i source and a PostgreSQL target:

Db2 for i Source Data Type	PostgreSQL Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 32766	bytea
char(*size*) for bit data, 1 <= size <= 32766	bytea
char(*size*), 1 <= size <= 32766	character varying(*size*), 1 <= size <= 32766
date	date
decfloat(*precision*), 16 <= p <= 34	character varying(255)
decimal(p,s), 1 <= p <= 63, 0 <= s <= 62	numeric(p,s), 1 <= p <= 63, 0 <= s <= 62
float	double precision
integer	integer

Db2 for i Source Data Type	PostgreSQL Target Data Type
long varbinary	bytea
long varchar	text
long varchar for bit data	bytea
numeric(p,s), 1 <= p <= 63, 0 <= s <= 62	numeric(p,s), 1 <= p <= 63, 0 <= s <= 62
real	real
rowid	bytea
smallint	smallint
time	time(0) without time zone
timestamp(*precision*), 0 <= p <= 6	timestamp(*precision*) without time zone, 0 <= p <= 6
timestamp(*precision*), 7 <= p <= 12	character varying(*size*), 27 <= size <= 32
varbinary(*size*), 1 <= size <= 32740	bytea
varchar(*size*) for bit data, 1 <= size <= 32740	bytea
varchar(*size*), 1 <= size <= 32740	character varying(*size*), 1 <= size <= 32740

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for i data types:

- BLOB
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for i Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for i source and a Snowflake target:

Db2 for i Source Data Type	Snowflake Target Data Type
bigint	integer
binary(*size*), 1 <= size <= 32766	binary(*size*), 1 <= size <= 32766
char(*size*) for bit data, 1 <= size <= 32766	binary(*size*), 1 <= size <= 32766
char(*size*), 1 <= size <= 32766	char(*size*), 4 <= size <= 131064
date	date
decfloat(*precision*), 16 <= p <= 34	char(255)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
decimal(p,s), 38 <= p <= 63, 0 <= s <= 62	char(*size*), 40 <= size <= 65
float	float
integer	integer
long varbinary	binary
long varchar	varchar
long varchar for bit data	binary
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
numeric(p,s), 38 <= p <= 63, 0 <= s <= 62	char(*size*), 40 <= size <= 65
real	float
rowid	binary(40)
smallint	integer
time	time(0)
timestamp(*precision*), 0 <= p <= 9	timestamp_ntz(*precision*), 0 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(*size*), 30 <= size <= 32
varbinary(*size*), 1 <= size <= 32740	binary(*size*), 1 <= size <= 32740
varchar(*size*) for bit data, 1 <= size <= 32740	binary(*size*), 1 <= size <= 32740
varchar(*size*), 1 <= size <= 32740	varchar(*size*), 4 <= size <= 130960

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for i data types:

- BLOB
- CLOB
- DATALINK
- DBCLOB
- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for LUW Source and Amazon Redshift Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for Linux, UNIX, and Windows (LUW) source and an Amazon Redshift target:

Db2 for LUW Source Data Type	Amazon Redshift Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 255	binary varying(*size*), 1 <= size <= 255
boolean	boolean
char for bit data	binary varying(1)
char(*size*) for bit data, 2 <= size <= 255	binary varying(*size*), 2 <= size <= 255
character(*size*), 1 <= s <= 255	character(*size*), 4 <= size <= 1020
date	date
decfloat(*precision*), 16 <= p <= 34	character varying(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	numeric(p,s), 1 <= p <= 31, 0 <= s <= 31
double	double precision
integer	integer
long varchar	character varying(65535)
long varchar for bit data	binary varying(32700)
real	real
smallint	smallint
time	time without time zone
timestamp(*precision*), 0 <= p <= 6	timestamp without time zone

Db2 for LUW Source Data Type	Amazon Redshift Target Data Type
timestamp(*precision*), 7 <= p <= 12	character varying(*size*), 27 <= size <= 32
varbinary(*size*), 1 <= size <= 32672	binary varying(*size*), 1 <= size <= 32672
varchar(*size*) for bit data, 1 <= size <= 32672	binary varying(*size*), 1 <= size <= 32672
varchar(*size*), 1 <= size <= 32672	character varying(*size*), 4 <= size <= 65535

Db2 for LUW Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for Linux, UNIX, and Windows (LUW) source and a Databricks target:

Db2 for LUW Source Data Type	Databricks Target Data Type
bigint	long
binary(*size*), 1 <= size <= 255	binary
boolean	boolean
char for bit data	binary
char(*size*) for bit data, 2 <= size <= 255	binary
character(*size*), 1 <= size <= 255	string
date	string
decfloat(*precision*), 16 <= p <= 34	string
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	decimal(p,s), 1 <= p <= 31, 0 <= s <= 31
double	double
integer	integer
long varchar	string
long varchar for bit data	binary
real	float
smallint	integer
time	string
timestamp(*precision*), 0 <= p <= 6	timestamp
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= size <= 32672	binary

Db2 for LUW Source Data Type	Databricks Target Data Type
varchar(*size*) for bit data, 1 <= size <= 32672	binary
varchar(*size*), 1 <= size <= 32672	string

Db2 for LUW Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for Linux, UNIX, and Windows (LUW) source and a Google BigQuery target:

Db2 for LUW Source Data Type	Google BigQuery Target Data Type
bigint	int64
binary(*size*), 1 <= size <= 255	bytes
boolean	bool
char for bit data	bytes
char(*size*) for bit data, 2 <= size <= 255	bytes
character(*size*), 1 <= size <= 255	string
date	date
decfloat(*precision*), 16 <= p <= 34	string
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric

Db2 for LUW Source Data Type	Google BigQuery Target Data Type
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(31,s), 10 <= s <= 31	bignumeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(5,s), 1 <= s <= 5	numeric
decimal(6,s), 1 <= s <= 6	numeric
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 30, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 31, 0 <= s <= 30	bignumeric
decimal(p,s), 9 <= p <= 31, 1 <= s <= 9	numeric
double	float64
integer	int64
long varchar	string
long varchar for bit data	bytes
real	float64
smallint	int64
time	time

Db2 for LUW Source Data Type	Google BigQuery Target Data Type
timestamp(*precision*), 0 <= p <= 6	datetime
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= s <= 32672	bytes
varchar(*size*) for bit data, 1 <= size <= 32672	bytes
varchar(*size*), 1 <= size <= 32672	string

Db2 for LUW Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for Linux, UNIX, and Windows (LUW) source and a Microsoft Azure Synapse Analytics target:

Db2 for LUW Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
blob	varbinary(max)
boolean	bit
char for bit data	binary
char(*size*) for bit data, 2 <= size <= 255	binary(*size*), 2 <= size <= 255
char(*size*), 1 <= size <= 255	char(*size*), 1 <= size <= 255
clob	varchar(max)
date	date
dbclob	varchar(max)
decfloat(*precision*), 16 <= p <= 34	varchar(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	decimal(p,s), 1 <= p <= 31, 0 <= s <= 31
double	float
integer	int
long varchar	varchar(max)
long varchar for bit data	varbinary(max)
long vargraphic	varchar(max)

Db2 for LUW Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
real	real
smallint	smallint
time	time(0)
timestamp(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
timestamp(*precision*), 8 <= p <= 12	char(*size*), 28 <= size <= 32
varbinary(*size*), 1 <= size <= 32672	varbinary(*size*), 1 <= size <= max
varchar(*size*) for bit data, 1 <= size <= 32672	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 32672	varchar(*size*), 1 <= size <= max
xml	varchar(max)

Db2 for LUW Source and Microsoft SQL Server Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for Linux, UNIX, and Windows (LUW) source and a Microsoft SQL Server target:

Db2 for LUW Source Data Type	Microsoft SQL Server Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 255	varbinary(*size*), 1 <= size <= 255
boolean	bit
char for bit data	varbinary
char(*size*) for bit data, 2 <= size <= 255	varbinary(*size*), 2 <= size <= 255
character(*size*), 1 <= size <= 255	varchar(*size*), 1 <= size <= 255
date	date
decfloat(*precision*), 16 <= p <= 34	varchar(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	decimal(p,s), 1 <= p <= 31, 0 <= s <= 31
double	float
integer	int
long varchar	varchar(max)
long varchar for bit data	varbinary(max)
real	real

Db2 for LUW Source Data Type	Microsoft SQL Server Target Data Type
smallint	smallint
time	time(0)
timestamp(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
timestamp(*precision*), 8 <= p <= 12	varchar(*size*), 28 <= size <= 32
varbinary(*size*), 1 <= size <= 32672	varbinary(*size*), 1 <= size <= max
varchar(*size*) for bit data, 1 <= size <= 32672	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 32672	varchar(*size*), 1 <= size <= max

Db2 for LUW Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for Linux, UNIX, and Windows (LUW) source and an Oracle target:

Db2 for LUW Source Data Type	Oracle Target Data Type
bigint	number(19)
binary(*size*), 1 <= size <= 255	raw(*size*), 1 <= size <= 255
boolean	char(1 char)
char for bit data	raw(1)
char(*size*) for bit data, 2 <= size <= 255	raw(*size*), 2 <= size <= 255
character(*size*), 1 <= size <= 255	varchar2(s byte), 1 <= s <= 255
date	date
decfloat(*precision*), 16 <= p <= 34	varchar2(255 char)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	number(p,s), 1 <= p <= 31, 0 <= s <= 31
double	binary_double
integer	number(10)
long varchar	clob
long varchar for bit data	blob
real	binary_float
smallint	number(5)
time	timestamp(0)

Db2 for LUW Source Data Type	Oracle Target Data Type
timestamp(0)	date
timestamp(*precision*), 1 <= p <= 9	timestamp(*precision*), 1 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(s char), 30 <= s <= 32
varbinary(*size*), 1 <= size <= 1501	raw(*size*), 1 <= size <= 1501
varbinary(*size*), 2001 <= size <= 32672	blob
varchar(*size*) for bit data, 1 <= size <= 1501	raw(*size*), 1 <= size <= 1501
varchar(*size*) for bit data, 2001 <= size <= 32672	blob
varchar(*size*), 1 <= size <= 3501	varchar2(s byte), 1 <= s <= 3501
varchar(*size*), 4001 <= size <= 32672	clob

Db2 for LUW Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for Linux, UNIX, and Windows (LUW) source and a Snowflake target:

Db2 for LUW Source Data Type	Snowflake Target Data Type
bigint	integer
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
blob	binary(*size*), 1 <= size <= max
boolean	boolean
char for bit data	binary(1)
char(*size*) for bit data, 2 <= size <= 255	binary(*size*), 2 <= size <= 255
character(*size*), 1 <= size <= 255	char(*size*), 4 <= size <= 1020
clob	varchar(*size*), 4 <= size <= max
date	date
dbclob	varchar(*size*), 4 <= size <= max
decfloat(*precision*), 16 <= p <= 34	char(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	number(p,s), 1 <= p <= 31, 0 <= s <= 31
double	float
integer	integer

Db2 for LUW Source Data Type	Snowflake Target Data Type
long varchar	varchar(*size*), 4 <= size <= 130800
long varchar for bit data	binary(*size*), 1 <= size <= 32700
long vargraphic	varchar(*size*), 4 <= size <= 65400
real	float
smallint	integer
time	time(0)
timestamp(*precision*), 0 <= p <= 9	timestamp_ntz(*precision*), 0 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(*size*), 30 <= size <= 32
varbinary(*size*), 1 <= size <= 32672	binary(*size*), 1 <= size <= 32672
varchar(*size*) for bit data, 1 <= size <= 32672	binary(*size*), 1 <= size <= 32672
varchar(*size*), 1 <= size <= 32672	varchar(*size*), 4 <= size <= 130688
xml	varchar(*size*), 4 <= size <= max

Db2 for z/OS Source and Amazon Redshift Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for z/OS source and an Amazon Redshift target:

Db2 for z/OS Source Data Type	Amazon Redshift Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 255	binary varying(*size*), 1 <= size <= 255
char for bit data	binary varying(1)
char(*size*) for bit data, 2 <= size <= 255	binary varying(*size*), 2 <= size <= 255
char(*size*), 1 <= size <= 255	character(*size*), 4 <= size <= 1020
date	date
decfloat(*precision*), 16 <= p <= 34	character varying(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	numeric(p,s), 1 <= p <= 31, 0 <= s <= 31
float	double precision
integer	integer
long varchar	character varying(65535)

Db2 for z/OS Source Data Type	Amazon Redshift Target Data Type
long varchar for bit data	binary varying(32704)
real	real
rowid	binary varying(40)
smallint	smallint
time	time without time zone
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp with time zone
timestamp(*precision*) with time zone, 7 <= p <= 12	character varying(*size*), 67 <= size <= 72
timestamp(*precision*), 0 <= p <= 6	timestamp without time zone
timestamp(*precision*), 7 <= p <= 12	character varying(*size*), 27 <= size <= 32
varbinary(*size*), 1 <= size <= 32704	binary varying(*size*), 1 <= size <= 32704
varchar(*size*) for bit data, 1 <= size <= 32704	binary varying(*size*), 1 <= size <= 32704
varchar(*size*), 1 <= size <= 32704	character varying(*size*), 4 <= size <= 65535

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for z/OS data types:

- BLOB
- CLOB
- DBCLOB
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for z/OS Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for z/OS source and a Databricks target:

Db2 for z/OS Source Data Type	Databricks Target Data Type
bigint	long
binary(*size*), 1 <= size <= 255	binary
char for bit data	binary
char(*size*) for bit data, 2 <= size <= 255	binary
char(*size*), 1 <= size <= 255	string

Db2 for z/OS Source Data Type	Databricks Target Data Type
date	string
decfloat(*precision*), 16 <= p <= 34	string
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	decimal(p,s), 1 <= p <= 31, 0 <= s <= 31
float	double
integer	integer
long varchar	string
long varchar for bit data	binary
real	float
rowid	binary
smallint	integer
time	string
timestamp(*precision*) with time zone, 0 <= p <= 12	string
timestamp(*precision*), 0 <= p <= 6	timestamp
timestamp(*precision*), 7 <= p <= 12	string
varbinary(*size*), 1 <= size <= 32704	binary
varchar(*size*) for bit data, 1 <= size <= 32704	binary
varchar(*size*), 1 <= size <= 32704	string

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for z/OS data types:

- BLOB
- CLOB
- DBCLOB
- DECIMAL with a scale greater than the precision
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for z/OS Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for z/OS source and a Google BigQuery target:

Db2 for z/OS Source Data Type	Google BigQuery Target Data Type
bigint	int64
binary(*size*), 1 <= size <= 255	bytes
char for bit data	bytes
char(*size*) for bit data, 2 <= size <= 255	bytes
char(*size*), 1 <= size <= 255	string
date	date
decfloat(*precision*), 16 <= p <= 34	string
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric

Db2 for z/OS Source Data Type	Google BigQuery Target Data Type
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(31,s), 10 <= s <= 31	bignumeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(5,s), 1 <= s <= 5	numeric
decimal(6,s), 1 <= s <= 6	numeric
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 30, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 31, 0 <= s <= 30	bignumeric
decimal(p,s), 9 <= p <= 31, 1 <= s <= 9	numeric
float	float64
integer	int64
long varchar	string
long varchar for bit data	bytes
real	float64
rowid	bytes
smallint	int64
time	time
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*) with time zone, 7 <= p <= 12	string
timestamp(*precision*), 0 <= p <= 6	datetime
timestamp(*precision*), 7 <= p <= 12	string

Db2 for z/OS Source Data Type	Google BigQuery Target Data Type
varbinary(*size*), 1 <= size <= 32704	bytes
varchar(*size*) for bit data, 1 <= size <= 32704	bytes
varchar(*size*), 1 <= size <= 32704	string

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for z/OS data types:

- BLOB
- CLOB
- DBCLOB
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for z/OS Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for z/OS source and a Microsoft Azure Synapse Analytics target:

Db2 for z/OS Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
char for bit data	binary
char(*size*) for bit data, 2 <= size <= 255	binary(*size*), 2 <= size <= 255
char(*size*), 1 <= size <= 255	char(*size*), 1 <= size <= 255
date	date
decfloat(*precision*), 16 <= p <= 34	varchar(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	decimal(p,s), 1 <= p <= 31, 0 <= s <= 31
float	float
integer	int
long varchar	varchar(max)
long varchar for bit data	varbinary(max)
real	real

Db2 for z/OS Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
rowid	varbinary(40)
smallint	smallint
time	time(0)
timestamp(*precision*) with time zone, 0 <= p <= 7	datetimeoffset(*precision*), 0 <= p <= 7
timestamp(*precision*) with time zone, 8 <= p <= 12	char(*size*), 68 <= size <= 72
timestamp(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
timestamp(*precision*), 8 <= p <= 12	char(*size*), 28 <= size <= 32
varbinary(*size*), 1 <= size <= 32704	varbinary(*size*), 1 <= size <= max
varchar(*size*) for bit data, 1 <= size <= 32704	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 32704	varchar(*size*), 1 <= size <= max

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for z/OS data types:

- BLOB
- CLOB
- DBCLOB
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for z/OS Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for z/OS source and an Oracle target:

Db2 for z/OS Source Data Type	Oracle Target Data Type
bigint	number(19)
binary(*size*), 1 <= size <= 255	raw(*size*), 1 <= size <= 255
char for bit data	blob
char(*size*) for bit data, 2 <= size <= 255	blob
char(*size*), 1 <= size <= 255	char(s byte), 1 <= s <= 255
date	date
decfloat(*precision*), 16 <= p <= 34	char(255 char)

Db2 for z/OS Source Data Type	Oracle Target Data Type
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	number(p,s), 1 <= p <= 31, 0 <= s <= 31
float	binary_double
integer	number(10)
long varchar	clob
long varchar for bit data	blob
real	binary_double
rowid	blob
smallint	number(5)
time	char(8 char)
timestamp(0)	date
timestamp(*precision*) with time zone, 0 <= p <= 9	timestamp(*precision*) with time zone, 0 <= p <= 9
timestamp(*precision*) with time zone, 10 <= p <= 12	char(s char), 70 <= s <= 72
timestamp(*precision*), 1 <= p <= 9	timestamp(*precision*), 1 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(s char), 30 <= s <= 32
varbinary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
varbinary(*size*), 2001 <= size <= 32704	blob
varchar(*size*) for bit data, 1 <= size <= 32704	blob
varchar(*size*), 1 <= size <= 3501	varchar2(s byte), 1 <= s <= 3501
varchar(*size*), 4001 <= size <= 32704	clob

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for z/OS data types:

- BLOB
- CLOB
- DBCLOB
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Db2 for z/OS Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Db2 for z/OS source and a Snowflake target:

Db2 for z/OS Source Data Type	Snowflake Target Data Type
bigint	integer
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
char for bit data	binary(1)
char(*size*) for bit data, 2 <= size <= 255	binary(*size*), 2 <= size <= 255
char(*size*), 1 <= size <= 255	char(*size*), 4 <= size <= 1020
date	date
decfloat(*precision*), 16 <= p <= 34	char(255)
decimal(p,s), 1 <= p <= 31, 0 <= s <= 31	number(p,s), 1 <= p <= 31, 0 <= s <= 31
float	float
integer	integer
long varchar	varchar(130816)
long varchar for bit data	binary(32704)
real	float
rowid	binary(40)
smallint	integer
time	time(0)
timestamp(*precision*) with time zone, 0 <= p <= 9	timestamp_tz(*precision*), 0 <= p <= 9
timestamp(*precision*) with time zone, 10 <= p <= 12	char(*size*), 70 <= s <= 72
timestamp(*precision*), 0 <= p <= 9	timestamp_ntz(*precision*), 0 <= p <= 9
timestamp(*precision*), 10 <= p <= 12	char(*size*), 30 <= size <= 32
varbinary(*size*), 1 <= size <= 32704	binary(*size*), 1 <= size <= 32704
varchar(*size*) for bit data, 1 <= size <= 32704	binary(*size*), 1 <= size <= 32704
varchar(*size*), 1 <= size <= 32704	varchar(*size*), 4 <= size <= 130816

Unsupported source data types

Database Ingestion and Replication does not support the following Db2 for z/OS data types:

- BLOB
- CLOB
- DBCLOB
- XML

Database ingestion and replication jobs propagate nulls for columns that have these data types.

Microsoft SQL Server or Azure SQL Database Source and Amazon Redshift Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Microsoft SQL Server or Azure SQL Database source and an Amazon Redshift target:

Microsoft SQL Server or Azure SQL Database Source Data Type	Amazon Redshift Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 8000	binary varying(*size*), 1 <= size <= 8000
bit	boolean
char(*size*), 1 <= size <= 8000	character varying(*size*), 1 <= size <= 8000
date	date
datetime	timestamp without time zone
datetime2(7)	character varying(27)
datetime2(*precision*), 0 <= p <= 6	timestamp without time zone
datetimeoffset(7)	character varying(67)
datetimeoffset(*precision*), 0 <= p <= 6	timestamp with time zone
decimal(38,38)	character varying(41)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
float	double precision
geography	binary varying(1024000)
geometry	binary varying(1024000)
hierarchyid	binary varying(892)
image	binary varying(1024000)
int	integer

Microsoft SQL Server or Azure SQL Database Source Data Type	Amazon Redshift Target Data Type
money	numeric(20,4)
nchar(*size*), 1 <= size <= 4000	character varying(*size*), 1 <= size <= 8000
ntext	character varying(65535)
numeric(38,38)	character varying(41)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
nvarchar(*size*), 1 <= size <= 4000	character varying(*size*), 1 <= size <= 8000
real	real
smalldatetime	timestamp without time zone
smallint	smallint
smallmoney	numeric(10,4)
sql_variant	binary varying(8016)
text	character varying(65535)
time(*precision*), 0 <= p <= 7	varchar(16)
timestamp(8)	binary varying(8)
tinyint	smallint
uniqueidentifier	character(36)
varbinary(*size*), 1 <= size <= 8000	binary varying(*size*), 1 <= size <= 8000
varchar(*size*), 1 <= size <= 8000	character varying(*size*), 1 <= size <= 8000
xml	character varying(65535)

LOB limitations

Database ingestion and replication initial load jobs can replicate data from SQL Server GEOGRAPHY, GEOMETRY, IMAGE, NTEXT, NVARCHAR(MAX), TEXT, VARBINARY(MAX), VARCHAR(MAX), and XML columns to Amazon Redshift targets. LOB data might be truncated before being written to the target. The truncation point depends on the data type and target type. For initial load jobs with an Amazon Redshift target, GEOGRAPHY, GEOMETRY, IMAGE and VARBINARY(MAX) data is truncated to 1024000 bytes, and NTEXT, NVARCHAR(MAX), TEXT, VARCHAR(MAX), and XML data is truncated to 65535 bytes. For more information, see the "Include LOBs" description in “Configuring the source” on page 106.

Unsupported source data types

None.

Microsoft SQL Server Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Microsoft SQL Server source and a Databricks target:

Microsoft SQL Server Source Data Type	Databricks Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 8000	binary
bit	boolean
char(*size*), 1 <= size <= 8000	string
date	string
datetime	timestamp
datetime2(7)	string
datetime2(*precision*), 0 <= p <= 6	timestamp
datetimeoffset(7)	string
datetimeoffset(*precision*), 0 <= p <= 6	timestamp
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
float	double
geography	binary
geometry	binary
hierarchyid	binary
image	binary
int	integer
money	decimal(19,4)
nchar(*size*), 1 <= size <= 4000	string
ntext	string
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
nvarchar(*size*), 1 <= size <= 4000	string
real	float
smalldatetime	timestamp
smallint	integer

Microsoft SQL Server Source Data Type	Databricks Target Data Type
smallmoney	decimal(10,4)
sql_variant	binary
text	string
time(*precision*), 0 <= p <= 7	string
timestamp(8)	binary
tinyint	integer
uniqueidentifier	string
varbinary(*size*), 1 <= size <= 8000	binary
varchar(*size*), 1 <= size <= 8000	string
xml	string

LOB limitations

Database ingestion and replication jobs that use any of the load types can replicate data from SQL Server GEOGRAPHY, GEOMETRY, IMAGE, NTEXT, NVARCHAR(MAX), TEXT, VARBINARY(MAX), VARCHAR(MAX), and XML columns to Databricks targets. LOB data might be truncated before being written to the target. The truncation point depends on the data type and load type. For initial load jobs with a Databricks target, the truncation point for all SQL Server LOB data types is 16777216 bytes. For incremental loads and combined loads, if LOB columns contain more than 8 KB of data, the data is truncated to 4000 bytes if stored inline or to approximately 8000 bytes if stored out-of-line. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Unsupported source data types

DECIMAL or NUMERIC types with a scale greater than the precision are not supported.

Microsoft SQL Server Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Microsoft SQL Server source and a Google BigQuery target:

Microsoft SQL Server Source Data Type	Google BigQuery Target Data Type
bigint	int64
binary(*size*), 1 <= size <= 8000	bytes
bit	bool
char(*size*), 1 <= size <= 8000	string
date	date

Microsoft SQL Server Source Data Type	Google BigQuery Target Data Type
datetime	datetime
datetime2(7)	string
datetime2(*precision*), 0 <= p <= 6	datetime
datetimeoffset(7)	string
datetimeoffset(*precision*), 0 <= p <= 6	timestamp
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric

Microsoft SQL Server Source Data Type	Google BigQuery Target Data Type
decimal(38,s), 10 <= s <= 38	bignumeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(5,s), 1 <= s <= 5	numeric
decimal(6,s), 1 <= s <= 6	numeric
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 30, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 31, 0 <= s <= 30	bignumeric
decimal(p,s), 31 <= p <= 32, 0 <= s <= 31	bignumeric
decimal(p,s), 32 <= p <= 33, 0 <= s <= 32	bignumeric
decimal(p,s), 33 <= p <= 34, 0 <= s <= 33	bignumeric
decimal(p,s), 34 <= p <= 35, 0 <= s <= 34	bignumeric
decimal(p,s), 35 <= p <= 36, 0 <= s <= 35	bignumeric
decimal(p,s), 36 <= p <= 37, 0 <= s <= 36	bignumeric
decimal(p,s), 37 <= p <= 38, 0 <= s <= 37	bignumeric
decimal(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
float	float64
geography	bytes
geometry	bytes
hierarchyid	bytes
image	bytes
int	int64
money	numeric
nchar(*size*), 1 <= size <= 4000	string
ntext	string

Microsoft SQL Server Source Data Type	Google BigQuery Target Data Type
numeric(1,1)	numeric
numeric(10,10)	bignumeric
numeric(11,s), 10 <= s <= 11	bignumeric
numeric(12,s), 10 <= s <= 12	bignumeric
numeric(13,s), 10 <= s <= 13	bignumeric
numeric(14,s), 10 <= s <= 14	bignumeric
numeric(15,s), 10 <= s <= 15	bignumeric
numeric(16,s), 10 <= s <= 16	bignumeric
numeric(17,s), 10 <= s <= 17	bignumeric
numeric(18,s), 10 <= s <= 18	bignumeric
numeric(19,s), 10 <= s <= 19	bignumeric
numeric(2,s), 1 <= s <= 2	numeric
numeric(20,s), 10 <= s <= 20	bignumeric
numeric(21,s), 10 <= s <= 21	bignumeric
numeric(22,s), 10 <= s <= 22	bignumeric
numeric(23,s), 10 <= s <= 23	bignumeric
numeric(24,s), 10 <= s <= 24	bignumeric
numeric(25,s), 10 <= s <= 25	bignumeric
numeric(26,s), 10 <= s <= 26	bignumeric
numeric(27,s), 10 <= s <= 27	bignumeric
numeric(28,s), 10 <= s <= 28	bignumeric
numeric(3,s), 1 <= s <= 3	numeric
numeric(38,s), 10 <= s <= 38	bignumeric
numeric(4,s), 1 <= s <= 4	numeric
numeric(5,s), 1 <= s <= 5	numeric
numeric(6,s), 1 <= s <= 6	numeric
numeric(7,s), 1 <= s <= 7	numeric

Microsoft SQL Server Source Data Type	Google BigQuery Target Data Type
numeric(8,s), 1 <= s <= 8	numeric
numeric(p,0), 1 <= p <= 18	int64
numeric(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
numeric(p,s), 29 <= p <= 30, 0 <= s <= 29	bignumeric
numeric(p,s), 30 <= p <= 31, 0 <= s <= 30	bignumeric
numeric(p,s), 31 <= p <= 32, 0 <= s <= 31	bignumeric
numeric(p,s), 32 <= p <= 33, 0 <= s <= 32	bignumeric
numeric(p,s), 33 <= p <= 34, 0 <= s <= 33	bignumeric
numeric(p,s), 34 <= p <= 35, 0 <= s <= 34	bignumeric
numeric(p,s), 35 <= p <= 36, 0 <= s <= 35	bignumeric
numeric(p,s), 36 <= p <= 37, 0 <= s <= 36	bignumeric
numeric(p,s), 37 <= p <= 38, 0 <= s <= 37	bignumeric
numeric(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
nvarchar(*size*), 1 <= size <= 4000	string
real	float64
smalldatetime	datetime
smallint	int64
smallmoney	numeric
sql_variant	bytes
text	string
time(7)	string
time(*precision*), 0 <= p <= 6	time
timestamp(8)	bytes
tinyint	int64
uniqueidentifier	string
varbinary(*size*), 1 <= size <= 8000	bytes

Microsoft SQL Server Source Data Type	Google BigQuery Target Data Type
varchar(*size*), 1 <= size <= 8000	string
xml	string

LOB limitations

Database ingestion and replication initial load jobs can replicate data from SQL Server GEOGRAPHY, GEOMETRY, IMAGE, NTEXT, NVARCHAR(MAX), TEXT, VARBINARY(MAX), VARCHAR(MAX), and XML columns to Google BigQuery targets. LOB data might be truncated before being written to the target. For all of the LOB data types, the truncation point is 8388608 bytes. For more information, see the "Include LOBs" description in “Configuring the source” on page 106.

Unsupported source data types

None.

Microsoft SQL Server Source and a Target That Uses the Parquet Output Format

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Microsoft SQL Server source and an Amazon S3, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, or Microsoft Fabric OneLake target that uses the Parquet output format:

Microsoft SQL Server Source Data Type	Parquet Data Type
bigint	init64
binary	byte array
bit	int32
char	string
date	string
datetime	string
datetime2	string
datetimeoffset	string
decimal	string
float	string
geography	byte array
geometry	byte array
hierarchyid	byte array
image	byte array

Microsoft SQL Server Source Data Type	Parquet Data Type
int	int32
money	string
nchar	string
ntext	string
numeric	string
nvarchar	string
real	string
smalldatetime	string
smallint	int32
smallmoney	string
sql_variant	byte array (initial load) string (incremental load)
text	string
time	string
timestamp	binary
tinyint	int32
uniqueidentifier	string
varbinary	byte array
varchar	string
xml	string

Microsoft SQL Server or Azure SQL Database Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Microsoft SQL Server or Azure SQL Database source and a Microsoft Azure Synapse Analytics target:

Microsoft SQL Server or Azure SQL Database Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
bigint	bigint
binary(*size*), 1 <= size <= 8000	binary(*size*), 1 <= size <= 8000
bit	bit
char(*size*), 1 <= size <= 8000	char(*size*), 1 <= size <= 8000
date	date
datetime	datetime2(3)
datetime2(*precision*), 0 <= p <= 7	datetime2(*precision*), 0 <= p <= 7
datetimeoffset(*precision*), 0 <= p <= 7	datetimeoffset(*precision*), 0 <= p <= 7
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
float	float
geography	varbinary(max)
geometry	varbinary(max)
hierarchyid	varbinary(892)
image	varbinary(max)
int	int
money	money
nchar(*size*), 1 <= size <= 4000	nchar(*size*), 1 <= size <= 4000
ntext	nvarchar(max)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	numeric(p,s), 1 <= p <= 38, 0 <= s <= 38
nvarchar(*size*), 1 <= size <= 4000	nvarchar(*size*), 1 <= size <= 4000
real	real
smalldatetime	datetime2(0)
smallint	smallint

Microsoft SQL Server or Azure SQL Database Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
smallmoney	smallmoney
sql_variant	varbinary(max)
text	varchar(max)
time(*precision*), 0 <= p <= 7	time(*precision*), 0 <= p <= 7
timestamp(8)	varbinary(8)
tinyint	tinyint
uniqueidentifier	uniqueidentifier
varbinary(*size*), 1 <= size <= 8000	varbinary(*size*), 1 <= size <= 8000
varchar(*size*), 1 <= size <= 8000	varchar(*size*), 1 <= size <= 8000
xml	varchar(max)

LOB limitations

Database ingestion and replication initial load jobs can replicate data from SQL Server GEOGRAPHY, GEOMETRY, IMAGE, NTEXT, NVARCHAR(MAX), TEXT, VARBINARY(MAX), VARCHAR(MAX), and XML columns to Azure Synapse Analytics targets. LOB data might be truncated before being written to the target. The truncation point depends on the data type and target type. For initial load jobs with an Azure Synapse Analytics target, GEOGRAPHY, GEOMETRY, IMAGE and VARBINARY(MAX) data is truncated to 1000000 bytes, and NTEXT, NVARCHAR(MAX), TEXT, VARCHAR(MAX), and XML data is truncated to 500000 bytes. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Unsupported source data types

None.

Microsoft SQL Server Source and Microsoft SQL Server Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Microsoft SQL Server source and a Microsoft SQL Server target:

SQL Server Source Data Type	SQL Server Target Data Type
bigint	bigint
binary	varbinary
bit	bit
char	varchar
date	date

SQL Server Source Data Type	SQL Server Target Data Type
datetime	datetime
datetime2	datetime2
datetimeoffset	datetimeoffset
decimal	decimal
float	float
geography	geography
geometry	geometry
hierarchyid	hierarchyid
image	image
int	int
money	money
nchar	nvarchar
ntext	ntext
numeric	numeric
nvarchar	nvarchar
real	real
smalldatetime	smalldatetime
smallint	smallint
smallmoney	smallmoney
sql_variant	sql_variant
text	text
time	time
timestamp	varbinary
tinyint	tinyint
uniqueidentifier	uniqueidentifier
varbinary	varbinary

SQL Server Source Data Type	SQL Server Target Data Type
varchar	varchar
xml	xml

Sql_variant target data type

Database ingestion and replication initial load jobs that have a SQL Server source and a SQL Server target and include a sql_variant source column convert the sql_variant data to hexadecimal format on the target. To convert data from hexadecimal format to varbinary format, run the following query:

```
SELECT <column_name>, CONVERT(varbinary,<column_name>) from <table_name>;
```

Replace <*column_name*> and <*table_name*> with the actual target column and table names.

LOB limitations

Database ingestion and replication initial load and incremental load jobs can replicate data from SQL Server GEOGRAPHY, GEOMETRY, IMAGE, NTEXT, NVARCHAR(MAX), TEXT, VARBINARY(MAX), VARCHAR(MAX), and XML columns to SQL Server targets. LOB data might be truncated before being written to the target. The truncation point depends on the data type and target type. For initial load jobs with a SQL Server target, GEOGRAPHY, GEOMETRY, IMAGE, TEXT, VARBINARY(MAX), and VARCHAR(MAX) data is truncated to 16777216 bytes, NTEXT and NVARCHAR(MAX) data to 33554432 bytes , and XML data to 33554442 bytes.

Unsupported source data types

None.

Microsoft SQL Server Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Microsoft SQL Server source and an Oracle target:

Microsoft SQL Server Source Data Type	Oracle Target Data Type
bigint	number(19)
binary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
binary(*size*), 2001 <= size <= 8000	blob
bit	char(1 char)
char(*size*), 1 <= size <= 2000	char(s char), 1 <= size <= 2000
char(*size*), 2001 <= size <= 4000	varchar2(s char), 2001 <= size <= 4000
char(*size*), 4001 <= size <= 8000	clob
date	date
datetime	timestamp(3)
datetime2(0)	date

Microsoft SQL Server Source Data Type	Oracle Target Data Type
datetime2(*precision*), 1 <= p <= 7	timestamp(*precision*), 1 <= p <= 7
datetimeoffset(*precision*), 0 <= p <= 7	timestamp(*precision*) with time zone, 0 <= p <= 7
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
float	binary_double
geography	blob
geometry	blob
hierarchyid	blob
image	blob
int	number(10)
money	number(19,4)
nchar(*size*), 1 <= size <= 1000	nchar(s char), 1 <= size <= 1000
nchar(*size*), 1001 <= size <= 2000	nvarchar2(s char), 1001 <=size <= 2000
nchar(*size*), 2001 <= size <= 4000	nclob
ntext	nclob
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
nvarchar(*size*), 1 <= size <= 2000	nvarchar2(s char), 1 <= size <= 2000
nvarchar(*size*), 2001 <= size <= 4000	nclob
real	binary_float
smalldatetime	date
smallint	number(5)
smallmoney	number(10,4)
sql_variant	blob
text	clob
time(*precision*), 0 <= p <= 7	timestamp(*precision*), 0 <= p <= 7
timestamp(8)	raw(8)
tinyint	number(3)
uniqueidentifier	char(36 char)

Microsoft SQL Server Source Data Type	Oracle Target Data Type
varbinary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
varbinary(*size*), 2001 <= size <= 8000	blob
varchar(*size*), 1 <= size <= 4000	varchar2(s char), 1 <= size <= 4000
varchar(*size*), 4001 <= size <= 8000	clob
xml	clob

LOB limitations

Database ingestion and replication initial load jobs can replicate data from SQL Server GEOGRAPHY, GEOMETRY, IMAGE, NTEXT, NVARCHAR(MAX), TEXT, VARBINARY(MAX), VARCHAR(MAX), and XML columns to Oracle targets. LOB data might be truncated before being written to the target. For all of the LOB data types, the truncation point is 16777216 bytes. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Unsupported source data types

None.

Microsoft SQL Server or Azure SQL Database Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Microsoft SQL Server or Azure SQL Database source and a Snowflake target:

Microsoft SQL Server or Azure SQL Database Source Data Type	Snowflake Target Data Type
bigint	number(38,0)
binary(*size*), 1 <= size <= 8000	binary(*size*), 1 <= size <= 8000
bit	boolean
char(*size*), 1 <= size <= 8000	varchar(*size*), 1 <= size <= 8000
date	date
datetime	timestamp_ntz(3)
datetime2(*precision*), 0 <= p <= 7	timestamp_ntz(7)
datetimeoffset(*precision*), 0 <= p <= 7	timestamp_tz(*precision*), 0 <= p <= 7
decimal(38,38)	char(41)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
float	float

Microsoft SQL Server or Azure SQL Database Source Data Type	Snowflake Target Data Type
geography	binary
geometry	binary
hierarchyid	binary(892)
image	binary
int	number(38,0)
money	number(19,4)
nchar(*size*), 1 <= size <= 4000	varchar(*size*), 4 <= size <= 16000
ntext	varchar
numeric(38,38)	char(41)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
nvarchar(*size*), 1 <= size <= 4000	varchar(*size*), 4 <= size <= 16000
real	float
smalldatetime	timestamp_ntz(0)
smallint	number(38,0)
smallmoney	number(10,4)
sql_variant	binary(8016)
text	varchar
time(*precision*), 0 <= p <= 7	time(*precision*), 0 <= p <= 7
timestamp(8)	binary(8)
tinyint	number(38,0)
uniqueidentifier	char(36)
varbinary(*size*), 1 <= size <= 8000	binary(*size*), 1 <= size <= 8000
varchar(*size*), 1 <= size <= 8000	varchar(*size*), 1 <= size <= 8000
xml	varchar

LOB limitations

Database ingestion and replication jobs that use any of the load types can replicate data from SQL Server GEOGRAPHY, GEOMETRY, IMAGE, NTEXT, NVARCHAR(MAX), TEXT, VARBINARY(MAX), VARCHAR(MAX), and XML columns to Snowflake targets. LOB data might be truncated before being written to the target. The truncation point depends on the data type and load type. For initial load jobs with a Snowflake target,

GEOGRAPHY, GEOMETRY, IMAGE and VARBINARY(MAX) data is truncated to 8388608 bytes, and NTEXT, NVARCHAR(MAX), TEXT, VARCHAR(MAX), and XML data is truncated to 16777216 bytes. For incremental loads and combined loads, if LOB columns contain more than 8 KB of data, the data is truncated to 4000 bytes if stored inline or to approximately 8000 bytes if stored out-of-line. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Unsupported source data types

For incremental load jobs and combined initial and incremental load jobs that have a Snowflake target, Database Ingestion and Replication does not support the SQL Server hierarchyid data type. Database ingestion and replication jobs will propagate nulls for columns that have these data types.

MongoDB Source and Amazon Redshift Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MongoDB source and an Amazon Redshift target:

MongoDB Source Data Type	Amazon Redshift Target Data Type
document(2147483647)	super
id(32)	character varying(128)

Note: For a MongoDB source, Database Ingestion and Replication creates two columns on the target: an ID column, which provides an identifier similar to a primary key, and a Document column, which contains a JSON object containing an entire MongoDB record. Therefore, the mappings show the target data type associated with each of these columns.

MongoDB Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MongoDB source and a Databricks target:

MongoDB Source Data Type	Databricks Target Data Type
document(2147483647)	string
id(32)	string

Note: For a MongoDB source, Database Ingestion and Replication creates two columns on the target: an ID column, which provides an identifier similar to a primary key, and a Document column, which contains a JSON object containing an entire MongoDB record. Therefore, the mappings show the target data type associated with each of these columns.

MongoDB Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MongoDB source and a Google BigQuery target:

MongoDB Source Data Type	Google BigQuery Target Data Type
document(2147483647)	string
id(32)	string

Note: For a MongoDB source, Database Ingestion and Replication creates two columns on the target: an ID column, which provides an identifier similar to a primary key, and a Document column, which contains a JSON object containing an entire MongoDB record. Therefore, the mappings show the target data type associated with each of these columns.

MongoDB Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MongoDB source and a Microsoft Azure Synapse Analytics target:

MongoDB Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
document(2147483647)	nvarchar(max)
id(32)	nchar(32)

Note: For a MongoDB source, Database Ingestion and Replication creates two columns on the target: an ID column, which provides an identifier similar to a primary key, and a Document column, which contains a JSON object containing an entire MongoDB record. Therefore, the mappings show the target data type associated with each of these columns.

MongoDB Source and Microsoft SQL Server Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MongoDB source and a Microsoft SQL Server target:

MongoDB Source Data Type	Microsoft SQL Server Target Data Type
document(2147483647)	varchar(max)
id(32)	char(128)

Note: For a MongoDB source, Database Ingestion and Replication creates two columns on the target: an ID column, which provides an identifier similar to a primary key, and a Document column, which contains a JSON object containing an entire MongoDB record. Therefore, the mappings show the target data type associated with each of these columns.

MongoDB Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MongoDB source and an Oracle target:

MongoDB Source Column Type	Oracle Target Data Type
document(2147483647)	clob
id(32)	char(128)

Note: For a MongoDB source, Database Ingestion and Replication creates two columns on the target: an ID column, which provides an identifier similar to a primary key, and a Document column, which contains a JSON object containing an entire MongoDB record. Therefore, the mappings show the target data type associated with each of these columns.

MongoDB Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MongoDB source and a Snowflake target:

MongoDB Source Data Type	Snowflake Target Data Type
document(2147483647)	variant
id(32)	varchar(128)

Note: For a MongoDB source, Database Ingestion and Replication creates two columns on the target: an ID column, which provides an identifier similar to a primary key, and a Document column, which contains a JSON object containing an entire MongoDB record. Therefore, the mappings show the target data type associated with each of these columns.

MySQL Source and Amazon Redshift Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MySQL source and an Amazon Redshift target:

MySQL Source Data Type	Amazon Redshift Target Data Type
bigint	bigint
bigint unsigned	numeric(20,0)
binary(*size*), 1 <= size <= 255	binary varying(*size*), 1 <= size <= 255
bit	binary varying(2)
bit(*precision*), 1 <= p <= 64	binary varying(*size*), 1 <= size <= 8
blob	binary varying(65535)
char(*size*), 1 <= size <= 255	character varying(*size*), 3 <= size <= 765

MySQL Source Data Type	Amazon Redshift Target Data Type
date	date
datetime	timestamp without time zone
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	numeric(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	character varying(*size*), 40 <= size <= 67
double	double precision
float	real
geomcollection	binary varying(1024000)
geometry	binary varying(1024000)
geometrycollection	binary varying(1024000)
int	integer
int unsigned	bigint
json	character varying(65535)
linestring	binary varying(1024000)
longblob	binary varying(1024000)
longtext	character varying(65535)
mediumblob	binary varying(1024000)
mediumint	integer
mediumint unsigned	integer
mediumtext	character varying(65535)
multilinestring	binary varying(1024000)
multipoint	binary varying(1024000)
multipolygon	binary varying(1024000)
numeric	numeric(10,0)
point	binary varying(1024000)
polygon	binary varying(1024000)
smallint	smallint
smallint unsigned	integer

MySQL Source Data Type	Amazon Redshift Target Data Type
text	character varying(65535)
time(*precision*), 0 <= p <= 6	character varying(*size*), 10 <= size <= 17
timestamp(*precision*), 0 <= p <= 6	timestamp without time zone
tinyblob	binary varying(255)
tinyint	smallint
tinyint unsigned	smallint
tinytext	character varying(255)
varbinary(*size*), 1 <= size <= 65535	binary varying(*size*), 1 <= size <= 65535
varchar(*size*), 1 <= size <= 21844	character varying(*size*), 3 <= size <= 65532
year	smallint

Unsupported source data types

Database Ingestion and Replication does not support the following MySQL data types:

- BLOB
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

If a database ingestion and replication task specifies a source schema that includes columns with the JSON data type, deployment of the task ignores the JSON columns and does not create corresponding columns on the target. For other unsupported data types, database ingestion and replication jobs propagate nulls, even though the data types appear in the default mappings.

MySQL Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MySQL source and a Databricks target:

MySQL Source Data Type	Databricks Target Data Type
bigint	long
bigint unsigned	decimal(20)

MySQL Source Data Type	Databricks Target Data Type
binary(*size*), 1 <= size <= 255	binary
bit(*precision*), 1 <= p <= 64	binary
blob	binary
char(*size*), 1 <= size <= 255	string
date	string
datetime	timestamp
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	decimal(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	string
double	double
float	float
geomcollection	binary
geometry	binary
geometrycollection	binary
int	integer
int unsigned	long
json	string
linestring	binary
longblob	binary
longtext	string
mediumblob	binary
mediumint	integer
mediumint unsigned	integer
mediumtext	string
multilinestring	binary
multipoint	binary
multipolygon	binary
numeric	decimal

MySQL Source Data Type	Databricks Target Data Type
point	binary
polygon	binary
smallint	integer
smallint unsigned	integer
text	string
time(*precision*), 0 <= p <= 6	string
timestamp(*precision*), 0 <= p <= 6	timestamp
tinyblob	binary
tinyint	integer
tinyint unsigned	integer
tinytext	string
varbinary(*size*), 1 <= size <= 65535	binary
varchar(*size*), 1 <= size <= 21844	string
year	integer

Unsupported source data types

Database Ingestion and Replication does not support the following MySQL data types:

- BLOB
- DECIMAL with a scale greater than the precision
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

If a database ingestion and replication task specifies a source schema that includes columns with the JSON data type, deployment of the task ignores the JSON columns and does not create corresponding columns on the target. For other unsupported data types, database ingestion and replication jobs propagate nulls, even though the data types appear in the default mappings.

MySQL Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MySQL source and a Google BigQuery target:

MySQL Source Data Type	Google BigQuery Target Data Type
bigint	int64
bigint unsigned	bignumeric
binary(*size*), 1 <= size <= 255	bytes
bit(*precision*), 1 <= p <= 64	bytes
blob	bytes
char(*size*), 1 <= size <= 255	string
date	date
datetime	datetime
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric

MySQL Source Data Type	Google BigQuery Target Data Type
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(40,0)	string
decimal(41,s), 0 <= s <= 1	string
decimal(42,s), 0 <= s <= 2	string
decimal(43,s), 0 <= s <= 3	string
decimal(44,s), 0 <= s <= 4	string
decimal(45,s), 0 <= s <= 5	string
decimal(46,s), 0 <= s <= 6	string
decimal(47,s), 0 <= s <= 7	string
decimal(48,s), 0 <= s <= 8	string
decimal(49,s), 0 <= s <= 9	string
decimal(5,s), 1 <= s <= 5	numeric
decimal(50,s), 0 <= s <= 10	string
decimal(51,s), 0 <= s <= 11	string
decimal(52,s), 0 <= s <= 12	string
decimal(53,s), 0 <= s <= 13	string
decimal(54,s), 0 <= s <= 14	string
decimal(55,s), 0 <= s <= 15	string
decimal(56,s), 0 <= s <= 16	string
decimal(57,s), 0 <= s <= 17	string
decimal(58,s), 0 <= s <= 18	string
decimal(59,s), 0 <= s <= 19	string

MySQL Source Data Type	Google BigQuery Target Data Type
decimal(6,s), 1 <= s <= 6	numeric
decimal(60,s), 0 <= s <= 20	string
decimal(61,s), 0 <= s <= 21	string
decimal(62,s), 0 <= s <= 22	string
decimal(63,s), 0 <= s <= 23	string
decimal(64,s), 0 <= s <= 24	string
decimal(65,s), 0 <= s <= 25	string
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 64, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 31, 0 <= s <= 29	bignumeric
decimal(p,s), 39 <= p <= 65, 21 <= s <= 29	bignumeric
decimal(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
double	float64
float	float64
geomcollection	bytes
geometry	bytes
geometrycollection	bytes
int	int64
int unsigned	int64
json	string
linestring	bytes
longblob	bytes
longtext	string
mediumblob	bytes

MySQL Source Data Type	Google BigQuery Target Data Type
mediumint	int64
mediumint unsigned	int64
mediumtext	string
multilinestring	bytes
multipoint	bytes
multipolygon	bytes
numeric	int64
point	bytes
polygon	bytes
smallint	int64
smallint unsigned	int64
text	string
time(*precision*), 0 <= p <= 6	string
timestamp(*precision*), 0 <= p <= 6	datetime
tinyblob	bytes
tinyint	int64
tinyint unsigned	int64
tinytext	string
varbinary(*size*), 1 <= size <= 65535	bytes
varchar(*size*), 1 <= size <= 21844	string
year	int64

Unsupported source data types

Database Ingestion and Replication does not support the following MySQL data types:

- BLOB
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT

- TEXT
- TINYBLOB
- TINYTEXT

If a database ingestion and replication task specifies a source schema that includes columns with the JSON data type, deployment of the task ignores the JSON columns and does not create corresponding columns on the target. For other unsupported data types, database ingestion and replication jobs propagate nulls, even though the data types appear in the default mappings.

MySQL Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MySQL source and a Microsoft Azure Synapse Analytics target:

MySQL Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
bigint	bigint
bigint unsigned	decimal(20)
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
bit	bit
bit(*precision*), 1 <= p <= 64	binary(*size*), 1 <= size <= 8
blob	varbinary(max)
char(*size*), 1 <= size <= 255	varchar(*size*), 4 <= size <= 1020
date	date
datetime	datetime2(0)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	decimal(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	char(*size*), 40 <= size <= 67
double	float
float	real
geomcollection	varbinary(max)
geometry	varbinary(max)
geometrycollection	varbinary(max)
int	int
int unsigned	bigint
json	varchar(max)
linestring	varbinary(max)

MySQL Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
longblob	varbinary(max)
longtext	varchar(max)
mediumblob	varbinary(max)
mediumint	int
mediumint unsigned	int
mediumtext	varchar(max)
multilinestring	varbinary(max)
multipoint	varbinary(max)
multipolygon	varbinary(max)
numeric	decimal(10)
point	varbinary(max)
polygon	varbinary(max)
smallint	smallint
smallint unsigned	int
text	varchar(max)
time(*precision*), 0 <= p <= 6	varchar(*size*), 10 <= size <= 17
timestamp(*precision*), 0 <= p <= 6	datetime2(*precision*), 0 <= p <= 6
tinyblob	binary(255)
tinyint	smallint
tinyint unsigned	smallint
tinytext	char(256)
varbinary(*size*), 1 <= size <= 65535	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 21844	varchar(*size*), 4 <= size <= max
year	smallint

Unsupported source data types

Database Ingestion and Replication does not support the following MySQL data types:

- BLOB
- JSON

- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

If a database ingestion and replication task specifies a source schema that includes columns with the JSON data type, deployment of the task ignores the JSON columns and does not create corresponding columns on the target. For other unsupported data types, database ingestion and replication jobs propagate nulls, even though the data types appear in the default mappings.

MySQL Source and Microsoft SQL Server Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MySQL source and a Microsoft SQL Server target:

MySQL Source Data Type	Microsoft SQL Server Target Data Type
bigint	bigint
bigint unsigned	decimal(20,0)
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
bit	bit
bit(*precision*), 1 <= p <= 64	binary(*size*), 1 <= size <= 8
blob	varbinary(max)
char(*size*), 1 <= size <= 255	varchar(*size*), 3 <= size <= 765
date	date
datetime	datetime2
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	decimal(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	char(*size*), 40 <= size <= 67
double	float
float	real
geomcollection	binary(8000)
geometry	varbinary(max)
geometrycollection	varbinary(max)
int	int

MySQL Source Data Type	Microsoft SQL Server Target Data Type
int unsigned	bigint
json	varchar(max)
linestring	varbinary(max)
longblob	varbinary(max)
longtext	varchar(max)
mediumblob	varbinary(max)
mediumint	int
mediumint unsigned	int
mediumtext	varchar(max)
multilinestring	varbinary(max)
multipoint	varbinary(max)
multipolygon	varbinary(max)
numeric(10,0)	decimal(10,0)
point	varbinary(max)
polygon	varbinary(max)
smallint	smallint
smallint unsigned	int
text	varchar(max)
time(*precision*), 0 <= p <= 6	varchar(*size*), 10 <= size <= 17
timestamp(*precision*), 0 <= p <= 3	datetime2
timestamp(*precision*), 4 <= p <= 6	datetime2(*precision*), 4 <= p <= 6
tinyblob	binary(255)
tinyint	smallint
tinyint unsigned	tinyint
tinytext	varchar(255)
varbinary(*size*), 1 <= size <= 65535	varbinary(*size*), 1 <= size <= max

MySQL Source Data Type	Microsoft SQL Server Target Data Type
varchar(*size*), 1 <= size <= 21844	varchar(*size*), 3 <= size <= max
year	smallint

Unsupported source data types

Database Ingestion and Replication does not support the following MySQL data types:

- BLOB
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

If a database ingestion and replication task specifies a source schema that includes columns with the JSON data type, deployment of the task ignores the JSON columns and does not create corresponding columns on the target. For other unsupported data types, database ingestion and replication jobs propagate nulls, even though the data types appear in the default mappings.

MySQL Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MySQL source and an Oracle target:

MySQL Source Data Type	Oracle Target Data Type
bigint	number(19)
bigint unsigned	number(20)
binary(*size*), 1 <= size <= 255	raw(*size*), 1 <= size <= 255
bit(*precision*), 1 <= p <= 64	raw(*size*), 1 <= size <= 8
blob	blob
char(*size*), 1 <= size <= 255	varchar2(s char), 3 <= s <= 765
date	date
datetime	date
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	number(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	char(s char), 40 <= s <= 67

MySQL Source Data Type	Oracle Target Data Type
double	binary_double
float	binary_float
geomcollection	blob
geometry	blob
geometrycollection	blob
int	number(10)
int unsigned	number(10)
json	clob
linestring	blob
longblob	blob
longtext	clob
mediumblob	blob
mediumint	number(7)
mediumint unsigned	number(8)
mediumtext	clob
multilinestring	blob
multipoint	blob
multipolygon	blob
numeric(10,0)	number(10)
point	blob
polygon	blob
smallint	number(5)
smallint unsigned	number(5)
text	clob
time(*precision*), 0 <= p <= 6	char(s char), 10 <= s <= 17
timestamp(*precision*), 0 <= p <= 6	timestamp(*precision*), 0 <= p <= 6
tinyblob	raw(255)

MySQL Source Data Type	Oracle Target Data Type
tinyint	number(3)
tinyint unsigned	number(3)
tinytext	varchar2(255 char)
varbinary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
varbinary(*size*), 2001 <= size <= 65535	long raw
varchar(*size*), 1 <= size <= 4000	varchar2(s char), 3 <= s <= 4000
varchar(*size*), 4001 <= size <= 21844	clob
year	number(4)

Unsupported source data types

Database Ingestion and Replication does not support the following MySQL data types:

- BLOB
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

If a database ingestion and replication task specifies a source schema that includes columns with the JSON data type, deployment of the task ignores the JSON columns and does not create corresponding columns on the target. For other unsupported data types, database ingestion and replication jobs propagate nulls, even though the data types appear in the default mappings.

MySQL Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a MySQL source and a Snowflake target:

MySQL Source Data Type	Snowflake Target Data Type
bigint	number(19)
bigint unsigned	number(20)
binary(*size*), 1 <= size <= 255	binary(*size*), 1 <= size <= 255
bit(*precision*), 1 <= p <= 64	binary(*size*), 1 <= size <= 8

MySQL Source Data Type	Snowflake Target Data Type
blob	binary(65535)
char(*size*), 1 <= size <= 255	varchar(*size*), 4 <= size <= 1020
date	date
datetime	timestamp_ntz(0)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 29	number(p,s), 1 <= p <= 38, 0 <= s <= 29
decimal(p,s), 39 <= p <= 65, 0 <= s <= 29	char(*size*), 40 <= size <= 67
double	float
float	float
geomcollection	binary
geometry	binary
geometrycollection	binary
int	number(10)
int unsigned	number(10)
linestring	binary
longblob	binary
longtext	varchar
mediumblob	binary
mediumint	number(7)
mediumint unsigned	number(8)
mediumtext	varchar
multilinestring	binary
multipoint	binary
multipolygon	binary
numeric(10,0)	integer
point	binary
polygon	binary
smallint	number(5)

MySQL Source Data Type	Snowflake Target Data Type
smallint unsigned	number(5)
text	varchar(87380)
time(*precision*), 0 <= p <= 6	varchar(*size*), 10 <= size <= 17
timestamp(*precision*), 0 <= p <= 6	timestamp_ntz(*precision*), 0 <= p <= 6
tinyblob	binary(255)
tinyint	number(3)
tinyint unsigned	number(3)
tinytext	varchar(340)
varbinary(*size*), 1 <= size <= 65535	binary(*size*), 1 <= size <= 65535
varchar(*size*), 1 <= size <= 21844	varchar(*size*), 4 <= size <= 87376
year	number(4)

Unsupported source data types

Database Ingestion and Replication does not support the following MySQL data types:

- BLOB
- JSON
- LONGBLOB
- LONGTEXT
- MEDIUMBLOB
- MEDIUMTEXT
- TEXT
- TINYBLOB
- TINYTEXT

If a database ingestion and replication task specifies a source schema that includes columns with the JSON data type, deployment of the task ignores the JSON columns and does not create corresponding columns on the target. For other unsupported data types, database ingestion and replication jobs propagate nulls, even though the data types appear in the default mappings.

Netezza Source and Amazon Redshift Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Netezza source and an Amazon Redshift target:

Netezza Source Data Type	Amazon Redshift Target Data Type
bigint	bigint
boolean	boolean
byteint	smallint
char(*size*), 1 <= size <= 4096	character(*size*), 1 <= size <= 4096
char(*size*), 4097 <= size <= 64000	character varying(*size*), 4097 <= size <= 64000
date	date
double	double precision
integer	integer
interval	character varying(55)
nchar(*size*), 1 <= size <= 16000	character varying(*size*), 4 <= size <= 64000
numeric(38,38)	character varying(41)
numeric(p), 1 <= p <= 38	numeric(p,0), 1 <= p <= 38
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	numeric(p,s), 1 <= p <= 38, 1 <= s <= 37
nvarchar(*size*), 1 <= size <= 16000	character varying(*size*), 4 <= size <= 64000
real	real
smallint	smallint
st_geometry(p), 1 <= p <= 63001	binary varying(*size*), 1 <= size <= 63001
time	time without time zone
timestamp	timestamp without time zone
timetz	timetz
varbinary(*size*), 1 <= size <= 64000	binary varying(*size*), 1 <= size <= 64000
varchar(*size*), 1 <= size <= 64000	character varying(*size*), 1 <= size <= 64000

Unsupported source data types

Database Ingestion and Replication does not support the following Netezza data type:

- ST_GEOMETRY

Note: Even though this data type appears in the default mappings, database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have this data type.

Netezza Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Netezza source and a Databricks target:

Netezza Source Data Type	Databricks Target Data Type
bigint	long
boolean	boolean
byteint	integer
char(*size*), 1 <= size <= 64000	string
date	string
double	double
integer	integer
interval	string
nchar(*size*), 1 <= size <= 16000	string
numeric(p,s), 1 <= p <= 38, 1 <= s <= 38	decimal(p,s), 1 <= p <= 38, 1 <= s <= 38
nvarchar(*size*), 1 <= size <= 16000	string
real	float
smallint	integer
st_geometry(*precision*), 1 <= p <= 63001	binary
time	string
timestamp	timestamp
timetz	string
varbinary(*size*), 1 <= size <= 64000	binary
varchar(*size*), 1 <= size <= 64000	string

Unsupported source data types

Database Ingestion and Replication does not support the following Netezza data type:

- NUMERIC with a scale greater than the precision
- ST_GEOMETRY

Note: Even though this data type appears in the default mappings, database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have this data type.

Netezza Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Netezza source and a Google BigQuery target:

Netezza Source Data Type	Google BigQuery Target Data Type
bigint	int64
boolean	bool
byteint	int64
char(*size*), 1 <= size <= 64000	string
date	date
double	float64
integer	int64
interval	string
nchar(*size*), 1 <= size <= 16000	string
numeric(10,10)	bignumeric
numeric(11,s), 10 <= s <= 11	bignumeric
numeric(12,s), 10 <= s <= 12	bignumeric
numeric(13,s), 10 <= s <= 13	bignumeric
numeric(14,s), 10 <= s <= 14	bignumeric
numeric(15,s), 10 <= s <= 15	bignumeric
numeric(16,s), 10 <= s <= 16	bignumeric
numeric(17,s), 10 <= s <= 17	bignumeric
numeric(18,s), 10 <= s <= 18	bignumeric
numeric(19,s), 10 <= s <= 19	bignumeric
numeric(20,s), 10 <= s <= 20	bignumeric
numeric(21,s), 10 <= s <= 21	bignumeric
numeric(22,s), 10 <= s <= 22	bignumeric
numeric(23,s), 10 <= s <= 23	bignumeric

Netezza Source Data Type	Google BigQuery Target Data Type
numeric(24,s), 10 <= s <= 24	bignumeric
numeric(25,s), 10 <= s <= 25	bignumeric
numeric(26,s), 10 <= s <= 26	bignumeric
numeric(27,s), 10 <= s <= 27	bignumeric
numeric(28,s), 10 <= s <= 28	bignumeric
numeric(29,s), 10 <= s <= 29	bignumeric
numeric(38,s), 10 <= s <= 38	bignumeric
numeric(*precision*), 1 <= p <= 17	int64
numeric(*precision*), 19 <= p <= 29	numeric
numeric(*precision*), 30 <= p <= 38	bignumeric
numeric(p,s), 1 <= p <= 38, 1 <= s <= 9	numeric
numeric(p,s), 30 <= p <= 31, 1 <= s <= 30	bignumeric
numeric(p,s), 31 <= p <= 32, 1 <= s <= 31	bignumeric
numeric(p,s), 32 <= p <= 33, 1 <= s <= 32	bignumeric
numeric(p,s), 33 <= p <= 34, 1 <= s <= 33	bignumeric
numeric(p,s), 34 <= p <= 35, 1 <= s <= 34	bignumeric
numeric(p,s), 35 <= p <= 36, 1 <= s <= 35	bignumeric
numeric(p,s), 36 <= p <= 37, 1 <= s <= 36	bignumeric
numeric(p,s), 37 <= p <= 38, 1 <= s <= 37	bignumeric
nvarchar(*size*), 1 <= size <= 16000	string
real	float64
smallint	int64
st_geometry(*precision*), 1 <= p <= 63001	bytes
time	time
timestamp	datetime
timetz	timestamp

Netezza Source Data Type	Google BigQuery Target Data Type
varbinary(*size*), 1 <= size <= 64000	bytes
varchar(*size*), 1 <= size <= 64000	string

Unsupported source data types

Database Ingestion and Replication does not support the following Netezza data type:

- ST_GEOMETRY

Note: Even though this data type appears in the default mappings, database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have this data type.

Netezza Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Netezza source and a Microsoft Azure Synapse Analytics target:

Netezza Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
bigint	bigint
boolean	bit
byteint	smallint
char(*size*), 1 <= size <= 8000	char(*size*), 1 <= size <= 8000
char(*size*), 9000 <= size <= 64000	nvarchar(max)
date	date
double	float
integer	int
interval	varchar(55)
nchar(*size*), 1 <= size <= 3100	nchar(*size*), 1 <= size <= 3100
nchar(*size*), 4100 <= size <= 16000	nvarchar(max)
numeric(p,s), 1 <= p <= 38, 1 <= s <= 38	decimal(p,s), 1 <= p <= 38, 1 <= s <= 38
nvarchar(*size*), 1 <= size <= 16000	nvarchar(*size*), 1 <= size <= max
real	real
smallint	smallint
st_geometry(*precision*), 1 <= p <= 63001	varbinary(*size*), 1 <= size <= max
time	time(6)

Netezza Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
timestamp	datetime2(6)
timetz	datetimeoffset(6)
varbinary(*size*), 1 <= size <= 64000	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 64000	varchar(*size*), 1 <= size <= max

Unsupported source data types

Database Ingestion and Replication does not support the following Netezza data type:

- ST_GEOMETRY

Note: Even though this data type appears in the default mappings, database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have this data type.

Netezza Source and Microsoft SQL Server Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Netezza source and a Microsoft SQL Server target:

Netezza Source Data Type	Microsoft SQL Server Target Data Type
bigint	bigint
boolean	bit
byteint	smallint
char(*size*), 1 <= size <= 64000	varchar(*size*), 1 <= size <= max
date	date
double	float
integer	int
interval	varchar(55)
nchar(*size*), 1 <= size <= 16000	nvarchar(*size*), 4 <= size <= max
numeric(p,s), 1 <= p <= 38, 1 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
nvarchar(*size*), 1 <= size <= 16000	nvarchar(*size*), 4 <= size <= max
real	real
smallint	smallint
st_geometry(*precision*), 1 <= p <= 63001	varbinary(*size*), 1 <= size <= max
time	time(6)

Netezza Source Data Type	Microsoft SQL Server Target Data Type
timestamp	datetime2(6)
timetz	datetimeoffset(6)
varbinary(*size*), 1 <= size <= 64000	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 64000	varchar(*size*), 1 <= size <= max

Unsupported source data types

Database Ingestion and Replication does not support the following Netezza data type:

- ST_GEOMETRY

Note: Even though this data type appears in the default mappings, database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have this data type.

Netezza Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Netezza source and an Oracle target:

Netezza Source Data Type	Oracle Target Data Type
bigint	number(19)
boolean	char(1 char)
byteint	number(3)
char(*size*), 1 <= size <= 3001	varchar2(s char), 1 <= s <= 3001
char(*size*), 4096 <= size <= 64000	clob
date	date
double	binary_double
integer	number(10)
interval	varchar2(55 char)
nchar(*size*), 1 <= size <= 901	nvarchar2(s char), 1 <= s <= 901
nchar(*size*), 1024 <= size <= 1100	char(2000 char)
nchar(*size*), 2100 <= size <= 16000	clob
numeric(p,s), 1 <= p <= 38, 1 <= s <= 38	number(p,s), 1 <= p <= 38, 1 <= s <= 38
nvarchar(1)	nvarchar2(1 char)
nvarchar(1001)	char(2000 char)

Netezza Source Data Type	Oracle Target Data Type
nvarchar(*size*), 2001 <= s <= 16000	clob
real	binary_float
smallint	number(5)
st_geometry(*precision*), 1 <= p <= 1001	raw(*size*), 1 <= size <= 1001
st_geometry(*precision*), 2001 <= p <= 63001	blob
time	timestamp(6)
timestamp	timestamp(6)
timetz	timestamp(*precision*) with time zone, 6 <= p <= null
varbinary(*size*), 1 <= size <= 1001	raw(*size*), 1 <= size <= 1001
varbinary(*size*), 2001 <= size <= 64000	blob
varchar(*size*), 1 <= size <= 3001	varchar2(s char), 1 <= s <= 3001
varchar(*size*), 4001 <= size <= 64000	clob

Unsupported source data types

Database Ingestion and Replication does not support the following Netezza data type:

- ST_GEOMETRY

Note: Even though this data type appears in the default mappings, database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have this data type.

Netezza Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Netezza source and a Snowflake target:

Netezza Source Data Type	Snowflake Target Data Type
bigint	integer
boolean	boolean
byteint	integer
char(*size*), 1 <= size <= 64000	char(*size*), 1 <= size <= 64000
date	date
double	float
integer	integer

Netezza Source Data Type	Snowflake Target Data Type
interval	varchar(55)
nchar(*size*), 1 <= size <= 16000	char(*size*), 4 <= size <= 64000
numeric(38,38)	char(41)
numeric(*precision*), 1 <= p <= 38	integer
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
nvarchar(*size*), 1 <= size <= 16000	varchar(*size*), 4 <= size <= 64000
real	float
smallint	integer
st_geometry(*precision*), 1 <= p <= 63001	binary(*size*), 1 <= size <= 63001
time	time(6)
timestamp	timestamp_ntz(6)
timetz	timestamp_tz(6)
varbinary(*size*), 1 <= size <= 64000	binary(*size*), 1 <= size <= 64000
varchar(*size*), 1 <= size <= 64000	varchar(*size*), 1 <= size <= 64000

Unsupported source data types

Database Ingestion and Replication does not support the following Netezza data type:

- ST_GEOMETRY

Note: Even though this data type appears in the default mappings, database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have this data type.

Oracle Source and Amazon Redshift Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an Oracle source and an Amazon Redshift target:

Oracle Source Data Type	Amazon Redshift Target Data Type
binary_double	float8
binary_float	float4
blob	binary varying(1024000)
char(s byte), 1 <= size <= 2000	character varying(*size*), 4 <= size <= 2000
char(s char), 1 <= size <= 2000	character varying(*size*), 4 <= size <= 8000

Oracle Source Data Type	Amazon Redshift Target Data Type
clob	character varying(65535)
date	timestamp without time zone
float(*precision*), 1 <= p <= 126	character varying(255)
integer	character varying(255)
long raw	varbyte
long(2147483648 byte)	character varying(65535)
nchar(s char), 1 <= size <= 2000	character varying(*size*), 4 <= size <= 8000
nclob	character varying(65535)
number	number
number(*,s), -84 <= s <= 127	character varying(255)
number(38,s), 0 <= s <= 37	numeric(38,s), 0 <= s <= 37
number(p,s), 1 <= p <= 38, -37 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
number(p,s), 1 <= p <= 38, -84 <= s <= 127	character varying(*size*), 40 <= size <= 130
number(p,s), 10 <= p <= 38, -28 <= s <= 37	numeric(p,s), 10 <= p <= 38, 0 <= s <= 37
number(p,s), 11 <= p <= 38, -27 <= s <= 37	numeric(p,s), 11 <= p <= 38, 0 <= s <= 37
number(p,s), 12 <= p <= 38, -26 <= s <= 37	numeric(p,s), 12 <= p <= 38, 0 <= s <= 37
number(p,s), 13 <= p <= 38, -25 <= s <= 37	numeric(p,s), 13 <= p <= 38, 0 <= s <= 37
number(p,s), 14 <= p <= 38, -24 <= s <= 37	numeric(p,s), 14 <= p <= 38, 0 <= s <= 37
number(p,s), 15 <= p <= 38, -23 <= s <= 37	numeric(p,s), 15 <= p <= 38, 0 <= s <= 37
number(p,s), 16 <= p <= 38, -22 <= s <= 37	numeric(p,s), 16 <= p <= 38, 0 <= s <= 37
number(p,s), 17 <= p <= 38, -21 <= s <= 37	numeric(p,s), 17 <= p <= 38, 0 <= s <= 37
number(p,s), 18 <= p <= 38, -20 <= s <= 37	numeric(p,s), 18 <= p <= 38, 0 <= s <= 37
number(p,s), 19 <= p <= 38, -19 <= s <= 37	numeric(p,s), 20 <= p <= 38, 0 <= s <= 37
number(p,s), 2 <= p <= 38, -36 <= s <= 37	numeric(p,s), 2 <= p <= 38, 0 <= s <= 37
number(p,s), 21 <= p <= 38, -17 <= s <= 37	numeric(p,s), 21 <= p <= 38, 0 <= s <= 37
number(p,s), 22 <= p <= 38, -16 <= s <= 37	numeric(p,s), 22 <= p <= 38, 0 <= s <= 37
number(p,s), 23 <= p <= 38, -15 <= s <= 37	numeric(p,s), 23 <= p <= 38, 0 <= s <= 37

Oracle Source Data Type	Amazon Redshift Target Data Type
number(p,s), 24 <= p <= 38, -14 <= s <= 37	numeric(p,s), 24 <= p <= 38, 0 <= s <= 37
number(p,s), 25 <= p <= 38, -13 <= s <= 37	numeric(p,s), 25 <= p <= 38, 0 <= s <= 37
number(p,s), 26 <= p <= 38, -12 <= s <= 37	numeric(p,s), 26 <= p <= 38, 0 <= s <= 37
number(p,s), 27 <= p <= 38, -11 <= s <= 37	numeric(p,s), 27 <= p <= 38, 0 <= s <= 37
number(p,s), 28 <= p <= 38, -10 <= s <= 37	numeric(p,s), 28 <= p <= 38, 0 <= s <= 37
number(p,s), 29 <= p <= 38, -9 <= s <= 37	numeric(p,s), 29 <= p <= 38, 0 <= s <= 37
number(p,s), 3 <= p <= 38, -35 <= s <= 37	numeric(p,s), 3 <= p <= 38, 0 <= s <= 37
number(p,s), 30 <= p <= 38, -8 <= s <= 37	numeric(p,s), 30 <= p <= 38, 0 <= s <= 37
number(p,s), 31 <= p <= 38, -7 <= s <= 37	numeric(p,s), 31 <= p <= 38, 0 <= s <= 37
number(p,s), 32 <= p <= 38, -6 <= s <= 37	numeric(p,s), 32 <= p <= 38, 0 <= s <= 37
number(p,s), 33 <= p <= 38, -5 <= s <= 37	numeric(p,s), 33 <= p <= 38, 0 <= s <= 37
number(p,s), 34 <= p <= 38, -4 <= s <= 37	numeric(p,s), 34 <= p <= 38, 0 <= s <= 37
number(p,s), 35 <= p <= 38, -3 <= s <= 37	numeric(p,s), 35 <= p <= 38, 0 <= s <= 37
number(p,s), 36 <= p <= 38, -2 <= s <= 37	numeric(p,s), 36 <= p <= 38, 0 <= s <= 37
number(p,s), 37 <= p <= 38, -1 <= s <= 37	numeric(p,s), 37 <= p <= 38, 0 <= s <= 37
number(p,s), 4 <= p <= 38, -34 <= s <= 37	numeric(p,s), 4 <= p <= 38, 0 <= s <= 37
number(p,s), 5 <= p <= 38, -33 <= s <= 37	numeric(p,s), 5 <= p <= 38, 0 <= s <= 37
number(p,s), 6 <= p <= 38, -32 <= s <= 37	numeric(p,s), 6 <= p <= 38, 0 <= s <= 37
number(p,s), 7 <= p <= 38, -31 <= s <= 37	numeric(p,s), 7 <= p <= 38, 0 <= s <= 37
number(p,s), 8 <= p <= 38, -30 <= s <= 37	numeric(p,s), 8 <= p <= 38, 0 <= s <= 37
number(p,s), 9 <= p <= 38, -29 <= s <= 37	numeric(p,s), 9 <= p <= 38, 0 <= s <= 37
nvarchar2(s char), 1 <= size <= 4000	character varying(*size*), 4 <= size <= 16000
raw(*size*), 1 <= size <= 2000	varbyte
rowid	character varying(18)
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp with time zone
timestamp(*precision*) with time zone, 7 <= p <= 9	character varying(29)
timestamp(*precision*), 1 <= p <= 6	timestamp without time zone

Oracle Source Data Type	Amazon Redshift Target Data Type
timestamp(*precision*), 7 <= p <= 9	character varying(*size*), 27 <= s <= 29
varchar2(s byte), 1 <= size <= 4000	character varying(*size*), 4 <= size <= 4000
varchar2(s char), 1 <= size <= 4000	character varying(*size*), 4 <= size <= 16000
xml	binary varying (1024000)

LOB limitations

Database ingestion and replication initial load, incremental load, and combined load jobs can replicate data from Oracle BLOB, CLOB, LONG, LONG RAW, NCLOB, and XML columns to Amazon Redshift targets. LOB column data might be truncated on the target. The truncation point depends on the data type and target type. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Columns that have the LONG, LONG RAW, and XML data types are supported in initial load jobs and in incremental load and combined load jobs that use the **Query-based** CDC method. However, jobs that use the **Log-based** CDC method do not replicate data from these columns to the generated target table.

Unsupported source data types

Database Ingestion and Replication does not support the following Oracle source data types:

- "ANY" types such as ANYTYPE, ANYDATA, ANYDATASET
- Extended types
- INTERVAL
- JSON
- BFILE type for storing LOB data externally
- UROWID
- Spatial types such as SDO_GEOMETRY
- User-defined types such as OBJECT, REF, VARRAY, nested table types

Source columns that have unsupported data types are excluded from the target definition.

Oracle Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Oracle source and a Databricks target:

Oracle Source Data Type	Databricks Target Data Type
binary_double	double
binary_float	float
blob	binary
char(s byte), 1 <= size <= 2000	string
char(s char), 1 <= size <= 2000	string

Oracle Source Data Type	Databricks Target Data Type
clob	string
date	timestamp
float(*precision*), 1 <= p <= 126	string
integer	string
long raw	binary
long(2147483648 byte)	string
nchar(s char), 1 <= size <= 2000	string
nclob	string
number	string
number(*,s), -84 <= s <= 127	string
number(p,s), 1 <= p <= 38, -37 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
number(p,s), 1 <= p <= 38, -84 <= s <= 127	string
nvarchar2(s char), 1 <= size <= 4000	string
raw(*size*), 1 <= size <= 2000	binary
rowid	string
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*) with time zone, 7 <= p <= 9	string
timestamp(*precision*), 1 <= p <= 6	timestamp
timestamp(*precision*), 7 <= p <= 9	string
varchar2(s byte), 1 <= size <= 4000	string
varchar2(s char), 1 <= size <= 4000	string
xml	binary

LOB limitations

Database ingestion and replication initial load, incremental load, and combined load jobs can replicate data from Oracle BLOB, CLOB, LONG, LONG RAW, NCLOB, and XML columns to Databricks targets. LOB column data might be truncated on the target. For all of the Oracle LOB data types replicated to this target type, the truncation point is 16777216 bytes. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Columns that have the LONG, LONG RAW, and XML data types are supported in initial load jobs and in incremental load and combined load jobs that use the **Query-based** CDC method. However, jobs that use the **Log-based** CDC method do not replicate data from these columns to the generated target table.

Unsupported source data types

Database Ingestion and Replication does not support the following Oracle source data types:

- "ANY" types such as ANYTYPE, ANYDATA, ANYDATASET
- Extended types
- INTERVAL
- JSON
- BFILE type for storing LOB data externally
- NUMBER with a scale greater than the precision
- UROWID
- Spatial types such as SDO_GEOMETRY
- User-defined types such as OBJECT, REF, VARRAY, nested table types

Source columns that have unsupported data types are excluded from the target definition.

Oracle Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Oracle source and a Google BigQuery target:

Oracle Source Data Type	Google BigQuery Target Data Type
binary_double	float64
binary_float	float64
blob	bytes
char(s byte), 1 <= size <= 2000	string
char(s char), 1 <= size <= 2000	string
clob	string
date	datetime
float(*precision*), 1 <= p <= 126	string
integer	string
long raw	bytes
long(2147483648 byte)	string
nchar(s char), 1 <= size <= 2000	string
nclob	string

Oracle Source Data Type	Google BigQuery Target Data Type
number	string
number(*,s), -84 <= s <= 127	string
number(1,-38)	bignumeric
number(1,s), -37 <= s <= -29	bignumeric
number(1,s), -84 <= s <= -39	string
number(10,-29)	bignumeric
number(11,-28)	bignumeric
number(12,-27)	bignumeric
number(13,-26)	bignumeric
number(14,-25)	bignumeric
number(15,-24)	bignumeric
number(16,-23)	bignumeric
number(17,-22)	bignumeric
number(18,-21)	bignumeric
number(19,-20)	bignumeric
number(2,-37)	bignumeric
number(20,-19)	bignumeric
number(21,-18)	bignumeric
number(22,-17)	bignumeric
number(23,-16)	bignumeric
number(24,-15)	bignumeric
number(25,-14)	bignumeric
number(26,-13)	bignumeric
number(27,-12)	bignumeric
number(28,-11)	bignumeric
number(29,-10)	bignumeric
number(3,-36)	bignumeric

Oracle Source Data Type	Google BigQuery Target Data Type
number(30,-9)	bignumeric
number(31,-8)	bignumeric
number(32,-7)	bignumeric
number(33,-6)	bignumeric
number(34,-5)	bignumeric
number(35,-4)	bignumeric
number(36,-3)	bignumeric
number(37,-2)	bignumeric
number(38,-1)	bignumeric
number(38,s), 10 <= s <= 38	bignumeric
number(4,-35)	bignumeric
number(5,-34)	bignumeric
number(6,-33)	bignumeric
number(7,-32)	bignumeric
number(8,-31)	bignumeric
number(9,-30)	bignumeric
number(p,s), 1 <= p <= 38, -28 <= s <= 9	numeric
number(p,s), 1 <= p <= 38, -36 <= s <= 38	bignumeric
number(p,s), 1 <= p <= 38, -84 <= s <= 127	string
number(p,s), 2 <= p <= 3, -35 <= s <= 38	bignumeric
number(p,s), 2 <= p <= 3, -84 <= s <= 127	string
nvarchar2(s char), 1 <= s <= 4000	string
raw(*size*), 1 <= size <= 2000	bytes
rowid	string
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*) with time zone, 7 <= p <= 9	string
timestamp(*precision*), 1 <= p <= 6	datetime

Oracle Source Data Type	Google BigQuery Target Data Type
timestamp(*precision*), 7 <= p <= 9	string
varchar2(s byte), 1 <= size <= 4000	string
varchar2(s char), 1 <= size <= 4000	string
xml	bytes(65535)

LOB limitations

Database ingestion and replication initial load, incremental load, and combined load jobs can replicate data from Oracle BLOB, CLOB, LONG, LONG RAW, NCLOB, and XML columns to Google BigQuery targets. LOB column data might be truncated on the target. For all of the Oracle LOB data types, the truncation point is 8388608 bytes. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Columns that have the LONG, LONG RAW, and XML data types are supported in initial load jobs and in incremental load and combined load jobs that use the **Query-based** CDC method. However, jobs that use the **Log-based** CDC method do not replicate data from these columns to the generated target table.

Unsupported source data types

Database Ingestion and Replication does not support the following Oracle source data types:

- "ANY" types such as ANYTYPE, ANYDATA, ANYDATASET
- Extended types
- INTERVAL
- JSON
- BFILE type for storing LOB data externally
- UROWID
- Spatial types such as SDO_GEOMETRY
- User-defined types such as OBJECT, REF, VARRAY, nested table types

Source columns that have unsupported data types are excluded from the target definition.

Oracle Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an Oracle source and a Microsoft Azure Synapse Analytics target:

Oracle Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
binary_double	float
binary_float	real
blob	varbinary(max)
char(s byte), 1 <= size <= 2000	varchar(*size*), 4 <= size <= 2000

Oracle Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
char(s char), 1 <= size <= 2000	varchar(*size*), 4 <= size <= 8000
clob	varchar(max)
date	datetime2(0)
float(*precision*), 1 <= p <= 126	varchar(255)
integer	varchar(255)
long raw	varbinary(max)
long(2147483648 byte)	varchar(max)
nchar(s char), 1 <= size <= 2000	nchar(*size*), 1 <= size <= 2000
nclob	nvarchar(max)
number	varchar(255)
number(*,s), -84 <= s <= 127	varchar(255)
number(p,s), 1 <= p <= 38, -37 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
number(p,s), 1 <= p <= 38, -84 <= s <= 127	varchar(*size*), 40 <= size <= 130
nvarchar2(s char), 1 <= s <= 4000	nvarchar(*size*), 1 <= size <= 4000
raw(*size*), 1 <= s <= 2000	varbinary(*size*), 1 <= size <= 2000
rowid	varchar(18)
timestamp(*precision*) with time zone, 0 <= p <= 7	datetimeoffset(*precision*), 0 <= p <= 7
timestamp(*precision*) with time zone, 8 <= p <= 9	char(*size*), 68 <= size <= 69
timestamp(*precision*), 1 <= p <= 7	datetime2(*precision*), 1 <= p <= 7
timestamp(*precision*), 8 <= p <= 9	char(*size*), 28 <= size <= 29
varchar2(s byte), 1 <= size <= 4000	varchar(*size*), 4 <= size <= 4000
varchar2(s char), 1 <= size <= 4000	varchar(*size*), 4 <= size <= max
xml	varbinary(max)

LOB limitations

Database ingestion and replication initial load, incremental load, and combined load jobs can replicate data from Oracle BLOB, CLOB, LONG, LONG RAW, NCLOB, and XML columns to Microsoft Azure Synapse Analytics targets. LOB column data might be truncated on the target. For an Azure Synapse Analytics target, BLOB and LONG RAW data is truncated to 1000000 bytes, and CLOB, LONG, NCLOB, and XML data is truncated to 500000 bytes. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Columns that have the LONG, LONG RAW, and XML data types are supported in initial load jobs and in incremental load and combined load jobs that use the **Query-based** CDC method. However, jobs that use the **Log-based** CDC method do not replicate data from these columns to the generated target table.

Unsupported source data types

Database Ingestion and Replication does not support the following Oracle source data types with any load type:

- "ANY" types such as ANYTYPE, ANYDATA, ANYDATASET
- Extended types
- INTERVAL
- JSON
- BFILE type for storing LOB data externally
- UROWID
- Spatial types such as SDO_GEOMETRY
- User-defined types such as OBJECT, REF, VARRAY, nested table types

Source columns that have unsupported data types are excluded from the target definition.

Oracle Source and Microsoft SQL Server Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an Oracle source and a SQL Server target:

Oracle Source Data Type	SQL Server Target Data Type
binary_double	float
binary_float	real
blob	varbinary
char	varchar
clob	varchar
date	datetime2
float	varchar
long raw	varbinary
long	varchar
nchar	nvarchar
nclob	nvarchar

Oracle Source Data Type	SQL Server Target Data Type
number	One of the following types: - decimal, 1 <= p <= 38 and s >= 0 and s <= p - money, s > p or s < 0, number of digits and scale can be accommodated - bigint, s <= 0, storage size can be accommodated - char
nvarchar2	nvarchar
raw	varbinary
rowid	varchar
timestamp	One of the following types: - datetime2, fraction seconds <=7 - char
timestamp with time zone	One of the following types: - datetimeoffset , fractional seconds <=7 - char
varchar2	varchar
xml	varbinary(max)

LOB limitations

Database ingestion and replication initial load, incremental load, and combined load jobs can replicate data from Oracle BLOB, CLOB, NCLOB, LONG, LONG RAW, and XML columns to SQL Server targets. LOB column data might be truncated on the target. The truncation point depends on the data type and target type. For all of the Oracle LOB data types and this target, the truncation point is 1677721 bytes. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Columns that have the LONG, LONG RAW, and XML data types are supported in initial load jobs and in incremental load and combined load jobs that use the **Query-based** CDC method. However, jobs that use the **Log-based** CDC method do not replicate data from these columns to the generated target table.

Unsupported source data types

Database Ingestion and Replication does not support the following Oracle source data types with any load type:

- "ANY" types such as ANYTYPE, ANYDATA, ANYDATASET
- Extended types
- INTERVAL
- JSON
- BFILE type for storing LOB data externally
- UROWID
- Spatial types such as SDO_GEOMETRY
- User-defined types such as OBJECT, REF, VARRAY, nested table types

Source columns that have unsupported data types are excluded from the target definition.

Oracle Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an Oracle source and an Oracle target:

Oracle Source Data Type	Oracle Target Data Type
binary_double	binary_double
binary_float	binary_float
blob	blob
char(s byte), 1 <= size <= 2000	char(s byte), 1 <= size <= 2000
char(s char), 1 <= size <= 2000	char(s char), 1 <= size <= 2000
clob	clob
date	date
float(*precision*), 1 <= p <= 126	float(*precision*), 1 <= p <= 126
integer	integer
long raw	long raw
long(2147483648 byte)	long(2147483648 byte)
nchar(s char), 1 <= size <= 2000	nchar(s char), 1 <= size <= 2000
nclob	nclob
number	number
number(*,s), -84 <= s <= 127	number(*,s), -84 <= s <= 127
number(p,s), 1 <= p <= 38, -84 <= s <= 127	number(p,s), 1 <= p <= 38, -84 <= s <= 127
nvarchar2(s char), 1 <= size <= 4000	nvarchar2(s char), 1 <= size <= 4000
raw(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
rowid	rowid
timestamp(*precision*) with time zone, 0 <= p <= 9	timestamp(*precision*) with time zone, 0 <= p <= 9
timestamp(*precision*), 1 <= p <= 9	timestamp(*precision*), 1 <= p <= 9
varchar2(s byte), 1 <= size <= 4000	varchar2(s byte), 1 <= size <= 4000
varchar2(s char), 1 <= size <= 4000	varchar2(s char), 1 <= size <= 4000
xml	blob

LOB limitations

Database ingestion and replication initial load, incremental load, and combined load jobs can replicate data from Oracle BLOB, CLOB, NCLOB, LONG, LONG RAW, and XML columns to Oracle targets. LOB column data might be truncated on the target. For all of the Oracle LOB data types and this target, the truncation point is 16777216 bytes. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Columns that have the LONG, LONG RAW, and XML data types are supported in initial load jobs and in incremental load and combined load jobs that use the **Query-based** CDC method. However, jobs that use the **Log-based** CDC method do not replicate data from these columns to the generated target table.

Unsupported source data types

Database Ingestion and Replication does not support the following Oracle source data types with any load type:

- "ANY" types such as ANYTYPE, ANYDATA, ANYDATASET
- Extended types
- INTERVAL
- JSON
- BFILE type for storing LOB data externally
- UROWID
- Spatial types such as SDO_GEOMETRY
- User-defined types such as OBJECT, REF, VARRAY, nested table types

Source columns that have unsupported data types are excluded from the target definition.

Oracle Source and PostgreSQL Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an Oracle source and a PostgreSQL target:

Oracle Source Data Type	PostgreSQL Target Data Type
binary_double	double precision
binary_float	real
blob	bytea
char(s byte), 1 <= s <= 2000	character varying(*size*), 1 <= size <= 500
char(s char), 1 <= s <= 2000	character varying(*size*), 1 <= size <= 2000
clob	text
date	timestamp(*precision*) without time zone, 0 <= p <= null
float(*precision*), 1 <= p <= 126	character varying(255)
integer	character varying(255)
long raw	bytea
long(2147483648 byte)	text

Oracle Source Data Type	PostgreSQL Target Data Type
nchar(s char), 1 <= s <= 2000	character varying(*size*), 1 <= s <= 2000
nclob	text
number	character varying(255)
number(*,s), -84 <= s <= 127	character varying(255)
number(p,s), 1 <= p <= 38, -84 <= s <= 127	numeric(p,s), 1 <= p <= 127, 0 <= s <= 127
nvarchar2(s char), 1 <= s <= 4000	character varying(*size*), 1 <= s <= 4000
raw(*size*), 1 <= s <= 2000	bytea
rowid	character(18)
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp(*precision*) with time zone, 0 <= p <= 6
timestamp(*precision*) with time zone, 7 <= p <= 9	character varying(*size*), 67 <= size <= 69
timestamp(*precision*), 1 <= p <= 6	timestamp(*precision*) without time zone, 1 <= p <= 6
timestamp(*precision*), 7 <= p <= 9	character varying(*size*), 27 <= size <= 29
varchar2(s byte), 1 <= s <= 4000	character varying(*size*), 1 <= size <= 1000
varchar2(s char), 1 <= s <= 4000	character varying(*size*), 1 <= size <= 4000
xml	bytea

LOB limitations

Database ingestion and replication initial load, incremental load, and combined load jobs can replicate data from Oracle BLOB, CLOB, NCLOB, LONG, LONG RAW, and XML columns to PostgreSQL targets. LOB column data might be truncated on the target. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Columns that have the LONG, LONG RAW, and XML data types are supported in initial load jobs and in incremental load and combined load jobs that use the **Query-based** CDC method. However, jobs that use the **Log-based** CDC method do not replicate data from these columns to the generated target table.

Unsupported source data types

Database Ingestion and Replication does not support the following Oracle source data types with any load type:

- "ANY" types such as ANYTYPE, ANYDATA, ANYDATASET
- Extended types
- INTERVAL
- JSON
- BFILE type for storing LOB data externally
- UROWID

- Spatial types such as SDO_GEOMETRY
- User-defined types such as OBJECT, REF, VARRAY, nested table types

Source columns that have unsupported data types are excluded from the target definition.

Oracle Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an Oracle source and a Snowflake target:

Oracle Source Data Type	Snowflake Target Data Type
binary_double	double
binary_float	double
blob	binary
char(s byte), 1 <= size <= 2000	varchar(4)
char(s char), 1 <= size <= 2000	varchar(4)
clob	varchar
date	timestamp_ntz(0)
float(*precision*), 1 <= p <= 126	varchar(255)
integer	varchar(255)
long raw	binary
long(2147483648 bytes)	varchar(65535)
nchar(s char), 1 <= size <= 2000	varchar(4)
nclob	varchar
number	number
number(*,s), -84 <= s <= 127	char(255)
number(p,s), 1 <= p <= 38, -37 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
number(p,s), 1 <= p <= 38, -84 <= s <= 127	char(*size*), 40 <= size <= 130
nvarchar2(s char), 1 <= size <= 4000	varchar(4)
raw(*size*), 1 <= size <= 2000	binary(*size*), 1 <= size <= 2000
rowid	varchar(18)
timestamp(*precision*) with local time zone, 0 <= p <= 9	timestampltz (*precision*), 0 <= p <= 9
timestamp(*precision*) with time zone, 0 <= p <= 9	timestamp_tz(*precision*), 0 <= p <= 9

Oracle Source Data Type	Snowflake Target Data Type
timestamp(*precision*), 1 <= p <= 9	timestamp_ntz(*precision*), 1 <= p <= 9
varchar2(s byte), 1 <= size <= 4000	varchar(*size*), 4 <= size <= 4000
varchar2(s char), 1 <= size <= 4000	varchar(*size*), 4 <= size <= 16000
xml	binary(8388608)

LOB limitations

Database ingestion and replication initial load, incremental load, and combined load jobs can replicate data from Oracle BLOB, CLOB, NCLOB, LONG, LONG RAW, and XML columns to Snowflake targets. LOB column data might be truncated on the target. For initial load jobs with a Snowflake target, BLOB and LONG RAW data is truncated to 8388608 bytes, and CLOB, LONG, NCLOB, and XML data is truncated to 16777216 bytes. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Columns that have the LONG, LONG RAW, and XML data types are supported in initial load jobs and in incremental load and combined load jobs that use the **Query-based** CDC method. However, jobs that use the **Log-based** CDC method do not replicate data from these columns to the generated target table.

Unsupported source data types

Database Ingestion and Replication does not support the following Oracle source data types with any load type:

- "ANY" types such as ANYTYPE, ANYDATA, ANYDATASET
- Extended types
- INTERVAL
- JSON
- BFILE type for storing LOB data externally
- UROWID
- Spatial types such as SDO_GEOMETRY
- User-defined types such as OBJECT, REF, VARRAY, nested table types

Source columns that have unsupported data types are excluded from the target definition.

PostgreSQL Source and Amazon Redshift Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a PostgreSQL source and an Amazon Redshift target:

PostgreSQL Source Data Type	Amazon Redshift Target Data Type
bigint	int8
bit varying(1)	boolean
bit varying(*precision*), 2 <= p <= 83886080	binary varying(*size*), 1 <= *size* <= 1024000
bit(1)	boolean

PostgreSQL Source Data Type	Amazon Redshift Target Data Type
bit(*precision*), 2 <= p <= 83886080	binary varying(*size*), 1 <= *size* <= 1024000
boolean	boolean
box	binary varying(115)
character varying(*size*), 1 <= size <= 10485760	character varying(*size*), 4 <= *size* <= 65535
character(*size*), 1 <= size <= 10485760	character varying(*size*), 4 <= *size* <= 65535
cidr	character varying(45)
circle	binary varying(87)
date	date
daterange	character varying(29)
double precision	double precision
inet	character varying(45)
int4range	character varying(25)
int8range	character varying(43)
integer	integer
json	super
jsonb	super
line	binary varying(85)
lseg	binary varying(117)
macaddr	character varying(17)
macaddr8	character varying(23)
money	numeric(20,2)
numeric	character varying(65535)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
numeric(p,s), 38 <= p <= 1000, 38 <= s <= 1000	character varying(*size*), 41 <= *size* <= 1003
numrange	character varying(65535)
path	binary varying(1024000)
point	binary varying(57)

PostgreSQL Source Data Type	Amazon Redshift Target Data Type
polygon	binary varying(1024000)
real	real
smallint	smallint
time(*precision*) with time zone, 0 <= p <= 6	timetz
time(*precision*) without time zone, 0 <= p <= 6	time
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp with time zone
timestamp(*precision*) without time zone, 0 <= p <= 6	timestamp
tsrange	character varying(63)
tstzrange	character varying(75)
uuid	character(36)
xml	character varying(65535)

LOB limitations

Database ingestion and replication initial load and incremental load jobs that use any of the load types can replicate data from PostgreSQL BYTEA, TEXT, XML, and other large-object columns to Amazon Redshift targets if you select the **Include LOBs** option under **Advanced** on the **Source** page of the task wizard. LOB column data is truncated before being written to the target if it is greater in size than a byte limit that depends on the LOB type. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Unsupported source data types

For initial load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types:

- ABSTIME
- Array types
- NAME
- Object identifier types
- PG_LSN
- RELTIME
- Text search types:
  - TSQUERY
  - TSVECTOR
- User-defined types

For incremental load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types, in addition to those not supported for initial load jobs:

- Spatial types
  - Box
  - Circle
  - Line
  - LSeg
  - Path
  - Point
  - Polygon
- Unbounded varying types

Database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have these data types.

PostgreSQL Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a PostgreSQL source and a Databricks target:

PostgreSQL Source Data Type	Databricks Target Data Type
bigint	bigint
bit varying(*precision*), 1 <= p <= 83886080	binary
bit(*precision*), 1 <= p <= 83886080	binary
boolean	boolean
box	binary
character varying(*size*), 1 <= size <= 10485760	string
character(*size*), 1 <= size <= 10485760	string
cidr	string
circle	binary
date	string
daterange	string
double precision	double
inet	string
int4range	string
int8range	string

PostgreSQL Source Data Type	Databricks Target Data Type
integer	int
json	string
jsonb	string
line	binary
lseg	binary
macaddr	string
macaddr8	string
money	decimal(19,2)
numeric	string
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 1000, 39 <= s <= 1000	string
numrange	string
path	binary
point	binary
polygon	binary
real	float
smallint	int
time(*precision*) with time zone, 0 <= p <= 6	string
time(*precision*) without time zone, 0 <= p <= 6	string
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*) without time zone, 0 <= p <= 6	timestamp
tsrange	string
tstzrange	string
uuid	string
xml	string

LOB limitations

Database ingestion and replication initial load and incremental load jobs can replicate data from PostgreSQL BYTEA, TEXT, XML, and other large-object columns to Databricks targets if you select the **Include LOBs**

option under **Advanced** on the **Source** page of the task wizard. LOB column data is truncated before being written to the target if it is greater in size than a byte limit that depends on the LOB type. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Unsupported source data types

For initial load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types:

- ABSTIME
- Array types
- NAME
- NUMERIC with a scale greater than the precision
- Object identifier types
- PG_LSN
- RELTIME
- Text search types:
  - TSQUERY
  - TSVECTOR
- User-defined types

For incremental load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types, in addition to those not supported for initial load jobs:

- Spatial types
  - Box
  - Circle
  - Line
  - LSeg
  - Path
  - Point
  - Polygon
- Unbounded varying types

Database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have these data types.

PostgreSQL Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a PostgreSQL source and a Google BigQuery target:

PostgreSQL Source Data Type	Google BigQuery Target Data Type
bigint	int64
bit varying(*precision*), 1 <= p <= 83886080	bytes
bit(*precision*), 1 <= p <= 83886080	bytes

PostgreSQL Source Data Type	Google BigQuery Target Data Type
boolean	bool
box	bytes
character varying(*size*), 1 <= size <= 10485760	string
character(*size*), 1 <= size <= 10485760	string
cidr	string
circle	bytes
date	date
daterange	string
double precision	float64
inet	string
int4range	string
int8range	string
integer	int64
json	string
jsonb	string
line	bytes
lseg	bytes
macaddr	string
macaddr8	string
money	numeric
numeric	string
numeric(1,1)	numeric
numeric(10,10)	bignumeric
numeric(11,s), 10 <= s <= 11	bignumeric
numeric(12,s), 10 <= s <= 12	bignumeric
numeric(13,s), 10 <= s <= 13	bignumeric
numeric(14,s), 10 <= s <= 14	bignumeric

PostgreSQL Source Data Type	Google BigQuery Target Data Type
numeric(15,s), 10 <= s <= 15	bignumeric
numeric(16,s), 10 <= s <= 16	bignumeric
numeric(17,s), 10 <= s <= 17	bignumeric
numeric(18,s), 10 <= s <= 18	bignumeric
numeric(19,s), 10 <= s <= 19	bignumeric
numeric(2,s), 1 <= s <= 2	numeric
numeric(20,s), 10 <= s <= 20	bignumeric
numeric(21,s), 10 <= s <= 21	bignumeric
numeric(22,s), 10 <= s <= 22	bignumeric
numeric(23,s), 10 <= s <= 23	bignumeric
numeric(24,s), 10 <= s <= 24	bignumeric
numeric(25,s), 10 <= s <= 25	bignumeric
numeric(26,s), 10 <= s <= 26	bignumeric
numeric(27,s), 10 <= s <= 27	bignumeric
numeric(28,s), 10 <= s <= 28	bignumeric
numeric(3,s), 1 <= s <= 3	numeric
numeric(38,s), 10 <= s <= 38	bignumeric
numeric(4,s), 1 <= s <= 4	numeric
numeric(5,s), 1 <= s <= 5	numeric
numeric(6,s), 1 <= s <= 6	numeric
numeric(7,s), 1 <= s <= 7	numeric
numeric(8,s), 1 <= s <= 8	numeric
numeric(*precision*), 1 <= p <= 18	int64
numeric(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
numeric(p,s), 29 <= p <= 30, 10 <= s <= 29	bignumeric
numeric(p,s), 30 <= p <= 31, 1 <= s <= 30	bignumeric
numeric(p,s), 31 <= p <= 32, 1 <= s <= 31	bignumeric

PostgreSQL Source Data Type	Google BigQuery Target Data Type
numeric(p,s), 32 <= p <= 33, 1 <= s <= 32	bignumeric
numeric(p,s), 33 <= p <= 34, 1 <= s <= 33	bignumeric
numeric(p,s), 34 <= p <= 35, 1 <= s <= 34	bignumeric
numeric(p,s), 35 <= p <= 36, 1 <= s <= 35	bignumeric
numeric(p,s), 36 <= p <= 37, 1 <= s <= 36	bignumeric
numeric(p,s), 37 <= p <= 38, 1 <= s <= 37	bignumeric
numeric(p,s), 39 <= p <= 1000, 39 <= s <= 1000	string
numeric(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
numrange	string
path	bytes
point	bytes
polygon	bytes
real	float64
smallint	int64
time(*precision*) with time zone, 0 <= p <= 6	string
time(*precision*) without time zone, 0 <= p <= 6	time
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*) without time zone, 0 <= p <= 6	datetime
tsrange	string
tstzrange	string
uuid	string
xml	string

LOB limitations

Database ingestion and replication initial load and incremental load jobs can replicate data from PostgreSQL BYTEA, TEXT, XML, and other large-object columns to Google BigQuery targets if you select the **Include LOBs** option under **Advanced** on the **Source** page of the task wizard. LOB column data is truncated before being written to the target if it is greater in size than a byte limit that depends on the LOB type. For more information, see the "Include LOBs" description in “Configuring the source” on page 106.

Unsupported source data types

For initial load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types:

- ABSTIME
- Array types
- NAME
- Object identifier types
- PG_LSN
- RELTIME
- Text search types:
  - TSQUERY
  - TSVECTOR
- User-defined types

For incremental load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types, in addition to those not supported for initial load jobs:

- Spatial types
  - Box
  - Circle
  - Line
  - LSeg
  - Path
  - Point
  - Polygon
- Unbounded varying types

Database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have these data types.

PostgreSQL Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a PostgreSQL source and a Microsoft Azure Synapse Analytics target:

PostgreSQL Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
bigint	bigint
bit varying(1)	varbinary(*size*), 1 <= size <= max
bit varying(*precision*), 2 <= p <= 83886080	varbinary(*size*), 1 <= size <= max
bit(1)	bit
bit(*precision*), 2 <= p <= 64000	binary(*size*), 1 <= size <= 8000
bit(*precision*), 64001 <= p <= 83886080	varbinary(max)

PostgreSQL Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
boolean	bit
box	varbinary(115)
character varying(*size*), 1 <= size <= 10485760	varchar(*size*), 4 <= size <= max
character(*size*), 1 <= size <= 10485760	varchar(*size*), 4 <= size <= max
cidr	varchar(45)
circle	varbinary(87)
date	date
daterange	varchar(29)
double precision	float
inet	varchar(45)
int4range	varchar(25)
int8range	varchar(43)
integer	int
json	varchar(max)
jsonb	varchar(max)
line	varbinary(85)
lseg	varbinary(117)
macaddr	varchar(17)
macaddr8	varchar(23)
money	decimal(19,2)
numeric	number
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 1000, 39 <= s <= 1000	varchar(*size*), 42 <= s <= 1003
numrange	varchar(max)
path	varbinary(max)
point	varbinary(57)
polygon	varbinary(max)

PostgreSQL Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
real	real
smallint	smallint
time(*precision*) with time zone, 0 <= p <= 6	datetimeoffset(*precision*), 0 <= p <= 6
time(*precision*) without time zone, 0 <= p <= 6	time(*precision*), 0 <= p <= 6
timestamp(*precision*) with time zone, 0 <= p <= 6	datetimeoffset(*precision*), 0 <= p <= 6
timestamp(*precision*) without time zone, 0 <= p <= 6	datetime2(*precision*), 0 <= p <= 6
tsrange	varchar(63)
tstzrange	varchar(75)
uuid	uniqueidentifier
xml	varchar(max)

LOB limitations

Database ingestion and replication initial load and incremental load jobs can replicate data from PostgreSQL BYTEA, TEXT, XML, and other large-object columns to Microsoft Azure Synapse Analytics targets if you select the **Include LOBs** option under **Advanced** on the **Source** page of the task wizard. LOB column data is truncated before being written to the target if it is greater in size than a byte limit that depends on the LOB type. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Unsupported source data types

For initial load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types:

- ABSTIME
- Array types
- NAME
- Object identifier types
- PG_LSN
- RELTIME
- Text search types:
  - TSQUERY
  - TSVECTOR
- User-defined types

For incremental load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types, in addition to those not supported for initial load jobs:

- Spatial types
  - Box
  - Circle

- Line
- LSeg
- Path
- Point
- Polygon

- Unbounded varying types

Database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have these data types.

PostgreSQL Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a PostgreSQL source and an Oracle target:

PostgreSQL Source Data Type	Oracle Target Data Type
bigint	number(19)
bit varying(*precision*), 1 <= p <= 83886080	blob
bit(*precision*), 1 <= p <= 83886080	blob
boolean	char(1 char)
box	blob
character varying(*size*), 1 <= s <= 1501	char(s char), 4 <= s <= 2000
character varying(*size*), 2001 <= s <= 10485760	clob
character(*size*), 1 <= s <= 2000	char(s char), 4 <= s <= 2000
character(*size*), 2100 <= s <= 10485760	clob
cidr	char(45 char)
circle	blob
date	date
daterange	char(29 char)
double precision	binary_double
inet	char(45 char)
int4range	char(25 char)
int8range	char(43 char)
integer	number(10)

PostgreSQL Source Data Type	Oracle Target Data Type
json	clob
jsonb	clob
line	blob
lseg	blob
macaddr	char(17 char)
macaddr8	char(23 char)
money	number(19,2)
numeric	clob
numeric(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
numeric(p,s), 39 <= p <= 1000, 39 <= s <= 1000	char(s char), 42 <= s <= 1003
numrange	clob
path	blob
point	blob
polygon	blob
real	binary_double
smallint	number(5)
time(*precision*) with time zone, 0 <= p <= 6	char(s char), 48 <= s <= 55
time(*precision*) without time zone, 0 <= p <= 6	timestamp(*precision*), 0 <= p <= 6
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp(*precision*) with time zone, 0 <= p <= 6
timestamp(*precision*) without time zone, 0 <= p <= 0	date
timestamp(*precision*) without time zone, 1 <= p <= 6	timestamp(*precision*), 1 <= p <= 6
tsrange	char(63 char)
tstzrange	char(75 char)
uuid	char(36 char)
xml	clob

LOB limitations

Database ingestion and replication initial load jobs can replicate data from PostgreSQL BYTEA, TEXT, XML, and other large-object columns to Oracle targets if you select the **Include LOBs** option under **Advanced** on

the **Source** page of the task wizard. LOB column data is truncated before being written to the target if it is greater in size than a byte limit that depends on the LOB type. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Unsupported source data types

For initial load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types:

- ABSTIME
- Array types
- NAME
- Object identifier types
- PG_LSN
- RELTIME
- Text search types:
  - TSQUERY
  - TSVECTOR
- User-defined types

For incremental load and initial and incremental load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types, in addition to those not supported for initial load jobs:

- Spatial types
  - Box
  - Circle
  - Line
  - LSeg
  - Path
  - Point
  - Polygon
- Unbounded varying types

Database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have these data types.

PostgreSQL Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a PostgreSQL source and a Snowflake target:

PostgreSQL Source Data Type	Snowflake Target Data Type
bigint	number
bit varying(*precision*), 1 <= p <= 83886080	binary(*size*), 1 <= size <= 6291457
bit(*precision*), 1 <= p <= 83886080	binary(*size*), 1 <= size <= 6291457
boolean	boolean

PostgreSQL Source Data Type	Snowflake Target Data Type
box	binary(115)
character varying(*size*), 1 <= size <= 10485760	varchar(*size*), 4 <= size <= 16776004
character(*size*), 1 <= size <= 10485760	varchar(*size*), 4 <= size <= 16776004
cidr	varchar(45)
circle	binary(87)
date	date
daterange	varchar(29)
double precision	float
inet	varchar(45)
int4range	varchar(25)
int8range	varchar(43)
integer	integer
json	variant
jsonb	variant
line	binary(85)
lseg	binary(117)
macaddr	varchar(17)
macaddr8	varchar(23)
money	number(19,2)
numeric	varchar(131074)
numeric(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
numeric(p,s), 38 <= p <= 1000, 38 <= s <= 1000	varchar(*size*), 41 <= size <= 1003
numrange	varchar(294917)
path	binary
point	binary(57)
polygon	binary
real	float

PostgreSQL Source Data Type	Snowflake Target Data Type
smallint	number
time(*precision*) with time zone, 0 <= p <= 6	timestamp_tz(*precision*), 0 <= p <= 6
time(*precision*) without time zone, 0 <= p <= 6	time(*precision*), 0 <= p <= 6
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp_tz(*precision*), 0 <= p <= 6
timestamp(*precision*) without time zone, 0 <= p <= 6	timestamp_ntz(*precision*), 0 <= p <= 6
tsrange	varchar(63)
tstzrange	varchar(75)
uuid	varchar
xml	variant

LOB limitations

Database ingestion and replication jobs that use any of the load types can replicate data from PostgreSQL BYTEA, TEXT, XML, and other large-object columns to Snowflake targets if you select the **Include LOBs** option under **Advanced** on the **Source** page of the task wizard. LOB column data is truncated before being written to the target if it is greater in size than a byte limit that depends on the LOB type. For more information, see the "Include LOBs" description in "Configuring the source" on page 106.

Unsupported source data types

For initial load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types:

- ABSTIME
- Array types
- NAME
- Object identifier types
- PG_LSN
- RELTIME
- Text search types:
  - TSQUERY
  - TSVECTOR
- User-defined types

For incremental load and initial and incremental load jobs, Database Ingestion and Replication does not support the following PostgreSQL data types, in addition to those not supported for initial load jobs:

- Spatial types
  - Box
  - Circle
  - Line

- LSeg
- Path
- Point
- Polygon

- Unbounded varying types

Database ingestion and replication jobs either fail to deploy or propagate nulls for columns that have these data types.

SAP HANA Source and Amazon Redshift Target

The following table identifies the default data-type mappings for Database Ingestion and Replication configurations with an SAP HANA or SAP HANA Cloud source and an Amazon Redshift target:

SAP HANA Source Data Type	Amazon Redshift Target Data Type
alphanum(*precision*), 1 <= p <= 127	character(*size*), 1 <= size <= 127
array	binary varying(1024000)
bigint	bigint
binary(*size*), 1 <= size <= 2000	binary varying(*size*), 1 <= size <= 2000
boolean	boolean
char(*size*), 1 <= size <= 2000	character(*size*), 1 <= size <= 2000
date	date
decimal	character varying(255)
decimal(38,38)	character varying(41)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
double	double precision
float	double precision
integer	integer
nchar(*size*), 1 <= size <= 2000	character varying(*size*), 4 <= size <= 8000
nvarchar(*size*), 1 <= size <= 5000	character varying(*size*), 4 <= size <= 20000
real	real
seconddate	timestamp without time zone
shorttext(*precision*), 1 <= p <= 5000	character varying(*size*), 1 <= size <= 5000
smalldecimal	character varying(255)

SAP HANA Source Data Type	Amazon Redshift Target Data Type
smallint	smallint
st_geometry	binary varying(1024000)
st_point	binary varying(1024000)
time	time without time zone
timestamp	character varying(27)
timestamp	timestamp without time zone
tinyint	smallint
varbinary(*size*), 1 <= size <= 5000	binary varying(*size*), 1 <= size <= 5000
varchar(*size*), 1 <= size <= 5000	character varying(*size*), 1 <= size <= 5000

Unsupported source data types:

- alphanum (SAP HANA Cloud only)
- array
- bintext
- blob
- clob
- char (SAP HANA Cloud only)
- nclob
- st_geometry
- st_point
- text

Database Ingestion and Replication replicates only nulls for columns that have these unsupported data types.

Note: Some unsupported data types might appear in the default mappings. However, nulls are replicated for these mappings.

SAP HANA Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an SAP HANA or SAP HANA Cloud source and a Databricks target:

SAP HANA Source Data Type	Databricks Target Data Type
alphanum(*precision*), 1 <= p <= 127	string
array	binary
bigint	long

SAP HANA Source Data Type	Databricks Target Data Type
binary(*size*), 1 <= size <= 2000	binary
boolean	boolean
char(*size*), 1 <= s <= 2000	string
date	string
decimal	string
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
double	double
float	double
integer	integer
nchar(*size*), 1 <= size <= 2000	string
nvarchar(*size*), 1 <= size <= 5000	string
real	float
seconddate	timestamp
shorttext(*precision*), 1 <= p <= 5000	string
smalldecimal	string
smallint	integer
st_geometry	binary
st_point	binary
time	string
timestamp	string
timestamp	timestamp
tinyint	integer
varbinary(*size*), 1 <= size <= 5000	binary
varchar(*size*), 1 <= size <= 5000	string

Unsupported source data types:

- alphanum (SAP HANA Cloud only)
- array
- bintext

- blob
- clob
- char (SAP HANA Cloud only)
- decimal with a scale greater than the precision
- nclob
- st_geometry
- st_point
- text

Database Ingestion and Replication replicates only nulls for columns that have these unsupported data types.

Note: Some unsupported data types might appear in the default mappings. However, nulls are replicated for these mappings.

SAP HANA Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an SAP HANA or SAP HANA Cloud source and a Google BigQuery target:

SAP HANA Source Data Type	Google BigQuery Target Data Type
alphanum(*precision*), 1 <= p <= 127	string
array	bytes
bigint	int64
binary(*size*), 1 <= size <= 2000	bytes
boolean	bool
char(*size*), 1 <= size <= 2000	string
date	date
decimal	string
decimal(1,1)	numeric
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric

SAP HANA Source Data Type	Google BigQuery Target Data Type
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(2,s), 1 <= s <= 2	numeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(3,s), 1 <= s <= 3	numeric
decimal(38,s), 10 <= s <= 38	bignumeric
decimal(4,s), 1 <= s <= 4	numeric
decimal(5,s), 1 <= s <= 5	numeric
decimal(6,s), 1 <= s <= 6	numeric
decimal(7,s), 1 <= s <= 7	numeric
decimal(8,s), 1 <= s <= 8	numeric
decimal(p,0), 1 <= p <= 18	int64
decimal(p,s), 19 <= p <= 29, 0 <= s <= 9	numeric
decimal(p,s), 29 <= p <= 30, 0 <= s <= 29	bignumeric
decimal(p,s), 30 <= p <= 31, 0 <= s <= 30	bignumeric
decimal(p,s), 31 <= p <= 32, 0 <= s <= 31	bignumeric
decimal(p,s), 32 <= p <= 33, 0 <= s <= 32	bignumeric
decimal(p,s), 33 <= p <= 34, 0 <= s <= 33	bignumeric

SAP HANA Source Data Type	Google BigQuery Target Data Type
decimal(p,s), 34 <= p <= 35, 0 <= s <= 34	bignumeric
decimal(p,s), 35 <= p <= 36, 0 <= s <= 35	bignumeric
decimal(p,s), 36 <= p <= 37, 0 <= s <= 36	bignumeric
decimal(p,s), 37 <= p <= 38, 0 <= s <= 37	bignumeric
decimal(p,s), 9 <= p <= 38, 1 <= s <= 9	numeric
double	float64
float	float64
integer	int64
nchar(*size*), 1 <= size <= 2000	string
nvarchar(*size*), 1 <= size <= 5000	string
real	float64
seconddate	datetime
shorttext(*precision*), 1 <= p <= 5000	string
smalldecimal	string
smallint	int64
st_geometry	bytes
st_point	bytes
time	time
timestamp	datetime
timestamp	string
tinyint	int64
varbinary(*size*), 1 <= size <= 5000	bytes
varchar(*size*), 1 <= size <= 5000	string

Unsupported source data types:

- alphanum (SAP HANA Cloud only)
- array
- bintext
- blob

- clob
- char (SAP HANA Cloud only)
- nclob
- st_geometry
- st_point
- text

Database Ingestion and Replication replicates only nulls for columns that have these unsupported data types.

Note: Some unsupported data types might appear in the default mappings. However, nulls are replicated for these mappings.

SAP HANA Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an SAP HANA or SAP HANA Cloud source and a Microsoft Azure Synapse Analytics target:

SAP HANA Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
alphanum(*precision*), 1 <= p <= 127	char(*size*), 1 <= size <= 127
array	varbinary(max)
bigint	bigint
binary(*size*), 1 <= size <= 2000	binary(*size*), 1 <= size <= 2000
boolean	bit
char(*size*), 1 <= size <= 2000	char(*size*), 1 <= size <= 2000
date	date
decimal	varchar(255)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
double	float
float	float
integer	int
nchar(*size*), 1 <= size <= 2000	char(*size*), 4 <= size <= 8000
nvarchar(*size*), 1 <= size <= 5000	varchar(*size*), 4 <= size <= max
real	real
seconddate	datetime2(0)
shorttext(*precision*), 1 <= p <= 5000	varchar(*size*), 1 <= s <= 5000

SAP HANA Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
smalldecimal	varchar(255)
smallint	smallint
st_geometry	varbinary(max)
st_point	varbinary(max)
time	time(0)
timestamp	datetime2(*precision*), 0 <= p <= 7
tinyint	tinyint
varbinary(*size*), 1 <= size <= 5000	varbinary(*size*), 1 <= size <= 5000
varchar(*size*), 1 <= size <= 5000	varchar(*size*), 1 <= size <= 5000

Unsupported source data types:

- alphanum (SAP HANA Cloud only)
- array
- bintext
- blob
- clob
- char (SAP HANA Cloud only)
- nclob
- st_geometry
- st_point
- text

Database Ingestion and Replication replicates only nulls for columns that have these unsupported data types.

Note: Some unsupported data types might appear in the default mappings. However, nulls are replicated for these mappings.

SAP HANA Source and Microsoft SQL Server Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an SAP HANA or SAP HANA Cloud source and a Microsoft SQL Server target:

SAP HANA Source Data Type	Microsoft SQL Server Target Data Type
alphanum(*precision*), 1 <= p <= 127	varchar(*size*), 1 <= size <= 127
array	varbinary(max)
bigint	bigint

SAP HANA Source Data Type	Microsoft SQL Server Target Data Type
binary(*size*), 1 <= size <= 2000	varbinary(*size*), 1 <= size <= 2000
boolean	bit
char(*size*), 1 <= size <= 2000	varchar(*size*), 1 <= size <= 2000
date	date
decimal	char(255)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	decimal(p,s), 1 <= p <= 38, 0 <= s <= 38
double	float
float	float
integer	int
nchar(*size*), 1 <= size <= 2000	nvarchar(*size*), 4 <= size <= 8000
nvarchar(*size*), 1 <= size <= 5000	nvarchar(*size*), 4 <= size <= max
real	real
real	real
seconddate	datetime2(0)
shorttext(*precision*), 1 <= p <= 5000	varchar(*size*), 1 <= size <= 5000
smalldecimal	char(255)
smallint	smallint
st_geometry	varbinary(max)
st_point	varbinary(max)
time	time(0)
timestamp	datetime2(*precision*), 0 <= p <= 7
tinyint	tinyint
varbinary(*size*), 1 <= size <= 5000	varbinary(*size*), 1 <= size <= 5000
varchar(*size*), 1 <= size <= 5000	varchar(*size*), 1 <= size <= 5000

Unsupported source data types:

- alphanum (SAP HANA Cloud only)
- array
- bintext

- blob
- clob
- char (SAP HANA Cloud only)
- nclob
- st_geometry
- st_point
- text

Database Ingestion and Replication replicates only nulls for columns that have these unsupported data types.

Note: Some unsupported data types might appear in the default mappings. However, nulls are replicated for these mappings.

SAP HANA Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an SAP HANA or SAP HANA Cloud source and an Oracle target:

SAP HANA Source Data Type	Oracle Target Data Type
alphanum(*precision*), 1 <= p <= 127	varchar2(s char), 1 <= s <= 127
array	blob
bigint	number(19)
binary(*size*), 1 <= size <= 2000	raw(*size*), 1 <= size <= 2000
boolean	char(1 char)
char(*size*), 1 <= size <= 2000	varchar2(s byte), 1 <= s <= 2000
date	date
decimal	char(255 char)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 38	number(p,s), 1 <= p <= 38, 0 <= s <= 38
double	binary_double
float	binary_double
integer	number(10)
nchar(*size*), 1 <= size <= 1000	nvarchar2(s char), 1 <= s <= 1000
nchar(*size*), 1001 <= size <= 2000	char(2000 char)
nvarchar(*size*), 1 <= size <= 901	nvarchar2(s char), 1 <= s <= 901
nvarchar(*size*), 1001 <= size <= 1901	char(2000 char)
nvarchar(*size*), 2001 <= size <= 5000	clob

SAP HANA Source Data Type	Oracle Target Data Type
real	binary_float
real	binary_float
seconddate	date
shorttext(*precision*), 1 <= p <= 3901	varchar2(s char), 1 <= s <= 3901
shorttext(*precision*), 4001 <= p <= 5000	clob
smalldecimal	char(255 char)
smallint	number(5)
st_geometry	blob
st_point	blob
time	timestamp(0)
timestamp	date
timestamp	timestamp(*precision*), 1 <= p <= 7
tinyint	number(3)
varbinary(*size*), 1 <= size <= 1501	raw(*size*), 1 <= size <= 1501
varchar(*size*), 1 <= size <= 3901	varchar2(s byte), 1 <= s <= 3901
varchar(*size*), 4001 <= size <= 5000	clob

Unsupported source data types:

- alphanum (SAP HANA Cloud only)
- array
- bintext
- blob
- clob
- char (SAP HANA Cloud only)
- nclob
- st_geometry
- st_point
- text

Database Ingestion and Replication replicates only nulls for columns that have these unsupported data types.

Note: Some unsupported data types might appear in the default mappings. However, nulls are replicated for these mappings.

SAP HANA Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with an SAP HANA or SAP HANA Cloud source and a Snowflake target:

SAP HANA Source Data Type	Snowflake Target Data Type
alphanum(*precision*), 1 <= p <= 127	char(*size*), 1 <= size <= 127
array	binary
bigint	integer
binary(*size*), 1 <= size <= 2000	binary(*size*), 1 <= size <= 2000
boolean	boolean
char(*size*), 1 <= size <= 2000	char(*size*), 1 <= size <= 2000
date	date
decimal	char(255)
decimal(38,38)	char(41)
decimal(p,s), 1 <= p <= 38, 0 <= s <= 37	number(p,s), 1 <= p <= 38, 0 <= s <= 37
double	float
float	float
integer	integer
nchar(*size*), 1 <= size <= 2000	char(*size*), 4 <= size <= 8000
nvarchar(*size*), 1 <= size <= 5000	varchar(*size*), 4 <= size <= 20000
real	float
seconddate	timestamp_ntz(0)
shorttext(*precision*), 1 <= p <= 5000	varchar(*size*), 1 <= size <= 5000
smalldecimal	char(255)
smallint	integer
st_geometry	binary
st_point	binary
time	time(0)
timestamp	timestamp_ntz(*precision*), 0 <= p <= 7
tinyint	integer

SAP HANA Source Data Type	Snowflake Target Data Type
varbinary(*size*), 1 <= size <= 5000	binary(*size*), 1 <= size <= 5000
varchar(*size*), 1 <= size <= 5000	varchar(*size*), 1 <= size <= 5000

Unsupported source data types:

- alphanum (SAP HANA Cloud only)
- array
- bintext
- blob
- clob
- char (SAP HANA Cloud only)
- nclob
- st_geometry
- st_point
- text

Database Ingestion and Replication replicates only nulls for columns that have these unsupported data types.

Note: Some unsupported data types might appear in the default mappings. However, nulls are replicated for these mappings.

Teradata Source and Amazon Redshift Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Teradata source and an Amazon Redshift target:

Teradata Source Data Type	Amazon Redshift Target Data Type
array	binary varying(1024000)
bigint	bigint
blob	binary varying(1024000)
byte(*precision*), 1 <= p <= 64000	binary varying(*size*), 1 <= size <= 64000
byteint	smallint
char(*size*), 1 <= size <= 64000	character varying(*size*), 4 <= size <= 65535
clob	character varying(65535)
date	date
decimal(p,s), 1 <= p <= 38, 1 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
float	double precision

Teradata Source Data Type	Amazon Redshift Target Data Type
integer	integer
interval day(*precision*) to hour, 1 <= p <= 4	character varying(*size*), 5 <= size <= 8
interval day(*precision*) to minute, 1 <= p <= 4	character varying(*size*), 8 <= size <= 11
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	character varying(*size*), 12 <= size <= 21
interval day(*precision*), 1 <= p <= 4	character varying(*size*), 2 <= size <= 5
interval hour(*precision*) to minute, 1 <= p <= 4	character varying(*size*), 5 <= size <= 8
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	character varying(*size*), 9 <= size <= 18
interval hour(*precision*), 1 <= p <= 4	character varying(*size*), 2 <= size <= 5
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	character varying(*size*), 9 <= size <= 18
interval minute(*precision*), 1 <= p <= 4	character varying(*size*), 2 <= size <= 5
interval month(*precision*), 1 <= p <= 4	character varying(*size*), 2 <= size <= 5
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	character varying(*size*), 3 <= size <= 12
interval year(*precision*) to month, 1 <= p <= 4	character varying(*size*), 5 <= size <= 8
interval year(*precision*), 1 <= p <= 4	character varying(*size*), 2 <= size <= 5
json	character varying(65535)
mbr	binary varying(256)
number(*,s), 0 <= s <= 37	character varying(255)
number(p,s), 1 <= p <= 38, 1 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	numeric(p,s), 1 <= p <= 38, 0 <= s <= 37
period(date)	character varying(28)
period(time(*precision*) with time zone), 0 <= p <= 6	character varying(*size*), 36 <= size <= 50
period(time(*precision*)), 0 <= p <= 6	character varying(*size*), 24 <= size <= 38
period(timestamp(*precision*) with time zone), 0 <= p <= 6	character varying(*size*), 58 <= size <= 72
period(timestamp(*precision*)), 0 <= p <= 6	character varying(*size*), 46 <= size <= 60
smallint	smallint
time(*precision*) with time zone, 0 <= p <= 6	timetz
time(*precision*), 0 <= p <= 6	time without time zone

Teradata Source Data Type	Amazon Redshift Target Data Type
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp with time zone
timestamp(*precision*), 0 <= p <= 6	timestamp without time zone
varbyte(*precision*), 1 <= p <= 64000	binary varying(*size*), 1 <= size <= 64000
varchar(*size*), 1 <= size <= 64000	character varying(*size*), 4 <= size <= 65535
varray	binary varying(1024000)
xml	character varying(65535)

Unsupported source data types

Database Ingestion and Replication does not support the following Teradata data types:

- ARRAY
- BLOB
- CLOB
- JSON
- ST_GEOMETRY
- XML

Note: Most of the unsupported data types appear in the default mappings. However, nulls are replicated for these mappings.

Teradata Source and Databricks Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Teradata source and a Databricks target:

Teradata Source Data Type	Databricks Target Data Type
array	binary
bigint	long
blob	binary
byte(*precision*), 1 <= p <= 64000	binary
byteint	integer
char(*size*), 1 <= size <= 64000	string
clob	string
date	string
decimal(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37

Teradata Source Data Type	Databricks Target Data Type
float	double
integer	integer
interval day(*precision*) to hour, 1 <= p <= 4	string
interval day(*precision*) to minute, 1 <= p <= 4	string
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval day(*precision*), 1 <= p <= 4	string
interval hour(*precision*) to minute, 1 <= p <= 4	string
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval hour(*precision*), 1 <= p <= 4	string
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval minute(*precision*), 1 <= p <= 4	string
interval month(*precision*), 1 <= p <= 4	string
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	string
interval year(*precision*) to month, 1 <= p <= 4	string
interval year(*precision*), 1 <= p <= 4	string
json	string
mbr	binary
number(*,s), 0 <= s <= 37	string
number(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
period(date)	string
period(time(*precision*) with time zone), 0 <= p <= 6	string
period(time(*precision*)), 0 <= p <= 6	string
period(timestamp(*precision*) with time zone), 0 <= p <= 6	string
period(timestamp(*precision*)), 0 <= p <= 6	string
smallint	integer
time(*precision*) with time zone, 0 <= p <= 6	string

Teradata Source Data Type	Databricks Target Data Type
time(*precision*), 0 <= p <= 6	string
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*), 0 <= p <= 6	timestamp
varbyte(*precision*), 1 <= p <= 64000	binary
varchar(*size*), 1 <= size <= 64000	string
varray	binary
xml	string

Unsupported source data types

Database Ingestion and Replication does not support the following Teradata data types:

- ARRAY
- BLOB
- CLOB
- DECIMAL, NUMBER, or NUMERIC with a scale greater than the precision
- JSON
- ST_GEOMETRY
- XML

Note: Most of the unsupported data types appear in the default mappings. However, nulls are replicated for these mappings.

Teradata Source and Google BigQuery Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Teradata source and a Google BigQuery target:

Teradata Source Data Type	Google BigQuery Target Data Type
array	bytes
bigint	int64
blob	bytes
byte(*precision*), 1 <= p <= 64000	bytes
byteint	int64
char(*size*), 1 <= size <= 64000	string
clob	string

Teradata Source Data Type	Google BigQuery Target Data Type
date	date
decimal(10,10)	bignumeric
decimal(11,s), 10 <= s <= 11	bignumeric
decimal(12,s), 10 <= s <= 12	bignumeric
decimal(13,s), 10 <= s <= 13	bignumeric
decimal(14,s), 10 <= s <= 14	bignumeric
decimal(15,s), 10 <= s <= 15	bignumeric
decimal(16,s), 10 <= s <= 16	bignumeric
decimal(17,s), 10 <= s <= 17	bignumeric
decimal(18,s), 10 <= s <= 18	bignumeric
decimal(19,s), 10 <= s <= 19	bignumeric
decimal(20,s), 10 <= s <= 20	bignumeric
decimal(21,s), 10 <= s <= 21	bignumeric
decimal(22,s), 10 <= s <= 22	bignumeric
decimal(23,s), 10 <= s <= 23	bignumeric
decimal(24,s), 10 <= s <= 24	bignumeric
decimal(25,s), 10 <= s <= 25	bignumeric
decimal(26,s), 10 <= s <= 26	bignumeric
decimal(27,s), 10 <= s <= 27	bignumeric
decimal(28,s), 10 <= s <= 28	bignumeric
decimal(29,s), 10 <= s <= 29	bignumeric
decimal(*precision*), 1 <= p <= 18	int64
decimal(*precision*), 19 <= p <= 29	numeric
decimal(*precision*), 30 <= p <= 38	bignumeric
decimal(p,s), 1 <= p <= 38, 1 <= s <= 9	numeric
decimal(p,s), 30 <= p <= 31, 1 <= s <= 30	bignumeric
decimal(p,s), 31 <= p <= 32, 1 <= s <= 31	bignumeric

Teradata Source Data Type	Google BigQuery Target Data Type
decimal(p,s), 32 <= p <= 33, 1 <= s <= 32	bignumeric
decimal(p,s), 33 <= p <= 34, 1 <= s <= 33	bignumeric
decimal(p,s), 34 <= p <= 35, 1 <= s <= 34	bignumeric
decimal(p,s), 35 <= p <= 36, 1 <= s <= 35	bignumeric
decimal(p,s), 36 <= p <= 37, 1 <= s <= 36	bignumeric
decimal(p,s), 37 <= p <= 38, 1 <= s <= 37	bignumeric
float	float64
integer	int64
interval day(*precision*) to hour, 1 <= p <= 4	string
interval day(*precision*) to minute, 1 <= p <= 4	string
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval day(*precision*), 1 <= p <= 4	string
interval hour(*precision*) to minute, 1 <= p <= 4	string
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval hour(*precision*), 1 <= p <= 4	string
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	string
interval minute(*precision*), 1 <= p <= 4	string
interval month(*precision*), 1 <= p <= 4	string
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	string
interval year(*precision*) to month, 1 <= p <= 4	string
interval year(*precision*), 1 <= p <= 4	string
json	string
mbr	bytes
number(*,s), 0 <= s <= 37	string
number(10,10)	bignumeric
number(11,s), 10 <= s <= 11	bignumeric
number(12,s), 10 <= s <= 12	bignumeric

Teradata Source Data Type	Google BigQuery Target Data Type
number(13,s), 10 <= s <= 13	bignumeric
number(14,s), 10 <= s <= 14	bignumeric
number(15,s), 10 <= s <= 15	bignumeric
number(16,s), 10 <= s <= 16	bignumeric
number(17,s), 10 <= s <= 17	bignumeric
number(18,s), 10 <= s <= 18	bignumeric
number(19,s), 10 <= s <= 19	bignumeric
number(20,s), 10 <= s <= 20	bignumeric
number(21,s), 10 <= s <= 21	bignumeric
number(22,s), 10 <= s <= 22	bignumeric
number(23,s), 10 <= s <= 23	bignumeric
number(24,s), 10 <= s <= 24	bignumeric
number(25,s), 10 <= s <= 25	bignumeric
number(26,s), 10 <= s <= 26	bignumeric
number(27,s), 10 <= s <= 27	bignumeric
number(28,s), 10 <= s <= 28	bignumeric
number(29,s), 10 <= s <= 29	bignumeric
number(*precision*), 1 <= p <= 18	int64
number(*precision*), 19 <= p <= 29	numeric
number(*precision*), 30 <= p <= 36	bignumeric
number(p,s), 1 <= p <= 38, 1 <= s <= 9	numeric
number(p,s), 30 <= p <= 31, 1 <= s <= 30	bignumeric
number(p,s), 31 <= p <= 32, 1 <= s <= 31	bignumeric
number(p,s), 32 <= p <= 33, 1 <= s <= 32	bignumeric
number(p,s), 33 <= p <= 34, 1 <= s <= 33	bignumeric
number(p,s), 34 <= p <= 35, 1 <= s <= 34	bignumeric
number(p,s), 35 <= p <= 36, 1 <= s <= 35	bignumeric

Teradata Source Data Type	Google BigQuery Target Data Type
number(p,s), 36 <= p <= 37, 1 <= s <= 36	bignumeric
number(p,s), 37 <= p <= 38, 1 <= s <= 37	bignumeric
numeric(10,10)	bignumeric
numeric(11,s), 10 <= s <= 11	bignumeric
numeric(12,s), 10 <= s <= 12	bignumeric
numeric(13,s), 10 <= s <= 13	bignumeric
numeric(14,s), 10 <= s <= 14	bignumeric
numeric(15,s), 10 <= s <= 15	bignumeric
numeric(16,s), 10 <= s <= 16	bignumeric
numeric(17,s), 10 <= s <= 17	bignumeric
numeric(18,s), 10 <= s <= 18	bignumeric
numeric(19,s), 10 <= s <= 19	bignumeric
numeric(20,s), 10 <= s <= 20	bignumeric
numeric(21,s), 10 <= s <= 21	bignumeric
numeric(22,s), 10 <= s <= 22	bignumeric
numeric(23,s), 10 <= s <= 23	bignumeric
numeric(24,s), 10 <= s <= 24	bignumeric
numeric(25,s), 10 <= s <= 25	bignumeric
numeric(26,s), 10 <= s <= 26	bignumeric
numeric(27,s), 10 <= s <= 27	bignumeric
numeric(28,s), 10 <= s <= 28	bignumeric
numeric(29,s), 10 <= s <= 29	bignumeric
numeric(*precision*), 1 <= p <= 18	int64
numeric(*precision*), 19 <= p <= 29	numeric
numeric(*precision*), 30 <= p <= 36	bignumeric
numeric(p,s), 1 <= p <= 38, 1 <= s <= 9	numeric
numeric(p,s), 30 <= p <= 31, 1 <= s <= 30	bignumeric

Teradata Source Data Type	Google BigQuery Target Data Type
numeric(p,s), 31 <= p <= 32, 1 <= s <= 31	bignumeric
numeric(p,s), 32 <= p <= 33, 1 <= s <= 32	bignumeric
numeric(p,s), 33 <= p <= 34, 1 <= s <= 33	bignumeric
numeric(p,s), 34 <= p <= 35, 1 <= s <= 34	bignumeric
numeric(p,s), 35 <= p <= 36, 1 <= s <= 35	bignumeric
numeric(p,s), 36 <= p <= 37, 1 <= s <= 36	bignumeric
numeric(p,s), 37 <= p <= 38, 1 <= s <= 37	bignumeric
period(date)	string
period(time(*precision*) with time zone), 0 <= p <= 6	string
period(time(*precision*)), 0 <= p <= 6	string
period(timestamp(*precision*) with time zone), 0 <= p <= 6	string
period(timestamp(*precision*)), 0 <= p <= 6	string
smallint	int64
time(*precision*) with time zone, 0 <= p <= 6	timestamp
time(*precision*), 0 <= p <= 6	time
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp
timestamp(*precision*), 0 <= p <= 6	datetime
varbyte(*precision*), 1 <= p <= 64000	bytes
varchar(*size*), 1 <= size <= 64000	string
varray	bytes
xml	string

Unsupported source data types

Database Ingestion and Replication does not support the following Teradata data types:

- ARRAY
- BLOB
- CLOB
- JSON
- ST_GEOMETRY
- XML

Note: Most of the unsupported data types appear in the default mappings. However, nulls are replicated for these mappings.

Teradata Source and Microsoft Azure Synapse Analytics Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Teradata source and a Microsoft Azure Synapse Analytics target:

Teradata Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
array	varbinary(max)
bigint	bigint
blob	varbinary(max)
byte(*precision*), 1 <= p <= 7501	binary(*size*), 1 <= size <= 7501
byte(*precision*), 8001 <= p <= 64000	varbinary(max)
byteint	smallint
char(*size*), 1 <= size <= 8000	char(*size*), 1 <= size <= 8000
char(*size*), 8001 <= size <= 64000	varchar(max)
clob	varchar(max)
date	date
decimal(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
float	float
integer	int
interval day(*precision*) to hour, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval day(*precision*) to minute, 1 <= p <= 4	varchar(*size*), 8 <= size <= 11
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 12 <= size <= 21
interval day(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval hour(*precision*) to minute, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 9 <= size <= 18
interval hour(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 9 <= size <= 18
interval minute(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5

Teradata Source Data Type	Microsoft Azure Synapse Analytics Target Data Type
interval month(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 3 <= size <= 12
interval year(*precision*) to month, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval year(*precision*), 1 <= p <= 4	varchar(*size*), 2 <=size <= 5
json	varchar(max)
mbr	varbinary(256)
number(*,s), 0 <= s <= 37	varchar(255)
number(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	decimal(p,s), 1 <= p <= 38, 1 <= s <= 37
period(date)	varchar(28)
period(time(*precision*) with time zone), 0 <= p <= 6	varchar(*size*), 36 <= size <= 50
period(time(*precision*)), 0 <= p <= 6	varchar(*size*), 24 <= size <= 38
period(timestamp(*precision*) with time zone), 0 <= p <= 6	varchar(*size*), 58 <= size <= 72
period(timestamp(*precision*)), 0 <= p <= 6	varchar(*size*), 46 <= size <= 60
smallint	smallint
time(*precision*) with time zone, 0 <= p <= 6	datetimeoffset(*precision*), 0 <= p <= 6
time(*precision*), 0 <= p <= 6	time(*precision*), 0 <= p <= 6
timestamp(*precision*) with time zone, 0 <= p <= 6	datetimeoffset(*precision*), 0 <= p <= 6
timestamp(*precision*), 0 <= p <= 6	datetime2(*precision*), 0 <= p <= 6
varbyte(*precision*), 1 <= p <= 64000	varbinary(*size*), 1 <= size <= max
varchar(*size*), 1 <= size <= 64000	varchar(*size*), 1 <= size <= max
varray	varbinary(max)
xml	varchar(max)

Unsupported source data types

Database Ingestion and Replication does not support the following Teradata data types:

- ARRAY
- BLOB
- CLOB

- JSON
- ST_GEOMETRY
- XML

Note: Most of the unsupported data types appear in the default mappings. However, nulls are replicated for these mappings.

Teradata Source and Oracle Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Teradata source and an Oracle target:

Teradata Source Data Type	Oracle Target Data Type
array	blob
bigint	number(19)
blob	blob
byte(*precision*), 1 <= p <= 64000	blob
byteint	number(3)
char(*size*), 1 <= size <= 2000	char(s byte), 1 <= s <= 2000
char(*size*), 2001 <= size <= 64000	clob
clob	clob
date	date
decimal(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
float	binary_double
integer	number(10)
interval day(*precision*) to hour, 1 <= p <= 4	char(s char), 5 <= s <= 8
interval day(*precision*) to minute, 1 <= p <= 4	char(s char), 8 <= s <= 11
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	char(s char), 12 <= s <= 21
interval day(*precision*), 1 <= p <= 4	char(s char), 2 <= s <= 5
interval hour(*precision*) to minute, 1 <= p <= 4	char(s char), 5 <= s <= 8
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	char(s char), 9 <= s <= 18
interval hour(*precision*), 1 <= p <= 4	char(s char), 2 <= s <= 5
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	char(s char), 9 <= s <= 18
interval minute(*precision*), 1 <= p <= 4	char(s char), 2 <= s <= 5

Teradata Source Data Type	Oracle Target Data Type
interval month(*precision*), 1 <= p <= 4	char(s char), 2 <= s <= 5
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	char(s char), 3 <= s <= 12
interval year(*precision*) to month, 1 <= p <= 4	char(s char), 5 <= s <= 8
interval year(*precision*), 1 <= p <= 4	char(s char), 2 <= s <= 5
json	clob
mbr	blob
number(*,s), 0 <= s <= 37	char(255 char)
number(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
period(date)	char(28 char)
period(time(*precision*) with time zone), 0 <= p <= 6	char(s char), 36 <= s <= 50
period(time(*precision*)), 0 <= p <= 6	char(s char), 24 <= s <= 38
period(timestamp(*precision*) with time zone), 0 <= p <= 6	char(s char), 58 <= s <= 72
period(timestamp(*precision*)), 0 <= p <= 6	char(s char), 46 <= s <= 60
smallint	number(5)
time(*precision*) with time zone, 0 <= p <= 6	timestamp(*precision*) with time zone, 0 <= p <= 6
time(*precision*), 0 <= p <= 6	timestamp(*precision*), 0 <= p <= 6
timestamp(0)	date
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp(*precision*) with time zone, 0 <= p <= 6
timestamp(*precision*), 1 <= p <= 6	timestamp(*precision*), 1 <= p <= 6
varbyte(*precision*), 1 <= p <= 64000	blob
varchar(*size*), 1 <= size <= 1501	char(s byte), 1 <= s <= 1501
varchar(*size*), 2001 <= size <= 64000	clob
varray	blob
xml	clob

Unsupported source data types

Database Ingestion and Replication does not support the following Teradata data types:

- ARRAY

- BLOB
- CLOB
- JSON
- ST_GEOMETRY
- XML

Note: Most of the unsupported data types appear in the default mappings. However, nulls are replicated for these mappings.

Teradata Source and Snowflake Target

The following table identifies the recommended data-type mappings for Database Ingestion and Replication configurations with a Teradata source and a Snowflake target:

Teradata Source Data Type	Snowflake Target Data Type
array	binary
bigint	integer
blob	binary
byte(*precision*), 1 <= p <= 64000	binary(*size*), 1 <= size <= 64000
byteint	integer
char(*size*), 1 <= size <= 64000	varchar(*size*), 4 <= size <= 256000
clob	varchar
date	date
decimal(*precision*), 1 <= p <= 38	integer
decimal(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
float	float
integer	integer
interval day(*precision*) to hour, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval day(*precision*) to minute, 1 <= p <= 4	varchar(*size*), 8 <= size <= 11
interval day(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 12 <= size <= 21
interval day(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval hour(*precision*) to minute, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval hour(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 9 <= size <= 18
interval hour(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5

Teradata Source Data Type	Snowflake Target Data Type
interval minute(*precision*) to second (s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 9 <= size <= 18
interval minute(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval month(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
interval second(p,s), 1 <= p <= 4, 0 <= s <= 6	varchar(*size*), 3 <= size <= 12
interval year(*precision*) to month, 1 <= p <= 4	varchar(*size*), 5 <= size <= 8
interval year(*precision*), 1 <= p <= 4	varchar(*size*), 2 <= size <= 5
json	varchar
mbr	binary(256)
number(*,s), 0 <= s <= 37	char(255)
number(*precision*), 1 <= p <= 36	integer
number(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
numeric(*precision*), 1 <= p <= 36	integer
numeric(p,s), 1 <= p <= 38, 1 <= s <= 37	number(p,s), 1 <= p <= 38, 1 <= s <= 37
period(date)	varchar(28)
period(time(*precision*) with time zone), 0 <= p <= 6	varchar(*size*), 36 <= size <= 50
period(time(*precision*)), 0 <= p <= 6	varchar(*size*), 24 <= size <= 38
period(timestamp(*precision*) with time zone), 0 <= p <= 6	varchar(*size*), 58 <= size <= 72
period(timestamp(*precision*)), 0 <= p <= 6	varchar(*size*), 46 <= size <= 60
smallint	integer
time(*precision*) with time zone, 0 <= p <= 6	timestamp_tz(*precision*), 0 <= p <= 6
time(*precision*), 0 <= p <= 6	time(*precision*), 0 <= p <= 6
timestamp(*precision*) with time zone, 0 <= p <= 6	timestamp_tz(*precision*), 0 <= p <= 6
timestamp(*precision*), 0 <= p <= 6	timestamp_ntz(*precision*), 0 <= p <= 6
varbyte(*precision*), 1 <= p <= 64000	binary(*size*), 1 <= size <= 64000
varchar(*size*), 1 <= size <= 64000	varchar(*size*), 4 <= size <= 256000
varray	binary
xml	varchar

Note: The Snowflake TIMESTAMP_TZ data type includes a default date that the source Teradata TIME WITH TIME ZONE data type does not include. For example, a database ingestion job will replicate the source value "12:59:59" as "1900-01-01 12:59:59".

Unsupported source data types

Database Ingestion and Replication does not support the following Teradata data types:

- ARRAY
- BLOB
- CLOB
- JSON
- ST_GEOMETRY
- XML

Note: Most of the unsupported data types appear in the default mappings. However, nulls are replicated for these mappings.

INDEX

F

G

I

M

N

O

P

R

S

T

U

W