Informatica® Cloud Data Integration

Snowflake Data Cloud Connector

Informatica Cloud Data Integration Snowflake Data Cloud Connector
July 2024

Table of Contents

Preface

Use *Snowflake Data Cloud Connector* to learn how to read from or write to Snowflake. Learn to create a connection, develop and run mappings, mapping tasks, dynamic mapping tasks, and data transfer tasks in Data Integration. Learn how to push down the transformation logic for processing to the Snowflake database.

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.

Part I: Getting Started with Snowflake Data Cloud Connector

This part contains the following chapters:

Introduction to Snowflake Data Cloud Connector

You can use Snowflake Data Cloud Connector to access Snowflake from Data Integration. You can read from or write to Snowflake and other third-party applications, databases, and flat files.

You can configure Snowflake Data Cloud Connector to read data from the following Snowflake objects:

- Snowflake tables and views, including Snowflake external tables, hybrid tables, and materialized views.
- Apache Iceberg tables that are managed by Snowflake or any external catalog.
- Snowflake that is enabled for staging data in Azure, Amazon, Google Cloud Platform, or Snowflake GovCloud.

You can write data to the following Snowflake objects:

- Snowflake tables and views.
- Apache Iceberg tables that are managed by Snowflake.
- Snowflake that is enabled for staging data in Azure, Amazon, Google Cloud Platform, or Snowflake GovCloud.

When you use Snowflake Data Cloud Connector, you can create a Snowflake Data Cloud connection and use the connection in Data Integration mappings and tasks. When you run a Snowflake Data Cloud mapping or task, the Secure Agent writes data to Snowflake based on the workflow and Snowflake Data Cloud connection configuration. You can create a mapping to read and write to a wide variety of heterogeneous data sources.

You can switch mappings to advanced mode to include transformations and functions that enable advanced functionality. A mapping in advanced mode can run on an advanced cluster hosted on Amazon Web Services, Google Cloud Platform, Microsoft Azure environment, or on a self-service cluster.

You can also create a mapping in SQL ELT mode to perform the data transformation entirely within your cloud ecosystem.

Preview Notice: The functionality to read from or write to Apache Iceberg tables is available for preview. Preview functionality is supported for evaluation purposes but is unwarranted and is not supported in production environments or any environment that you plan to push to production. Informatica intends to include the preview functionality in an upcoming release for production use, but might choose not to in accordance with changing market or technical circumstances. Note that if you are working on a preview POD, all data is excluded from SOC 2 compliance coverage. For more information, contact Informatica Global Customer Support.

Snowflake Data Cloud Connector assets

Create assets in Data Integration to integrate data using Snowflake Data Cloud Connector.

When you use Snowflake Data Cloud Connector, you can include the following Data Integration assets:

- Data transfer task

- Dynamic mapping task

- Mapping. For more information on mappings, see Chapter 3, "Mappings for Snowflake Data Cloud" on page 28.

- Mapping - SQL ELT. For more information on mappings in SQL ELT mode, see Chapter 10, "Mappings in SQL ELT mode for Snowflake Data Cloud" on page 76.

- Mapping task

For more information about configuring assets and transformations, see *Mappings*, *Transformations*, and *Tasks* in the Data Integration documentation.

C H A P T E R   2

Connections for Snowflake Data Cloud

Create a Snowflake Data Cloud connection to securely read data from or write data to Snowflake. You can use a Snowflake Data Cloud connection to specify sources, targets, and lookups in mappings and mapping tasks. You can also use the Snowflake connection in an SQL transformation.

Prepare for authentication

You can configure standard, authorization code, key pair, and client credentials authentication types to access Snowflake.

Before you configure the connection properties, you need to keep the authentication details handy based on the authentication type that you want to use.

Standard

To connect to Snowflake using standard authentication, you need the Snowflake account user name and password.

Let's get the required details such as the Snowflake account name, warehouse, and role details from the Snowflake account.

The following image shows you where you can find the name of your Snowflake account:
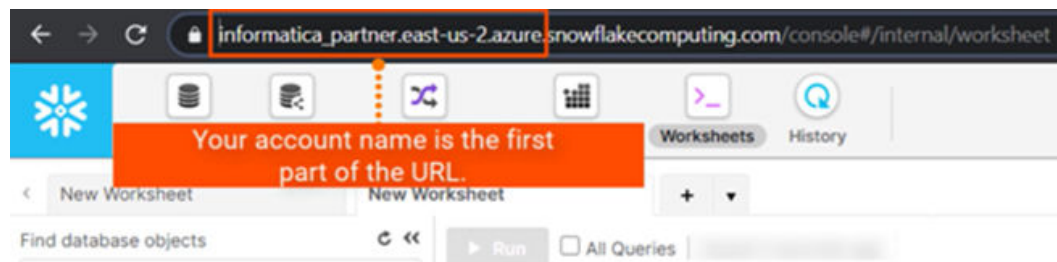
The following image shows you where you can find the name of the warehouse and role details of your Snowflake account:
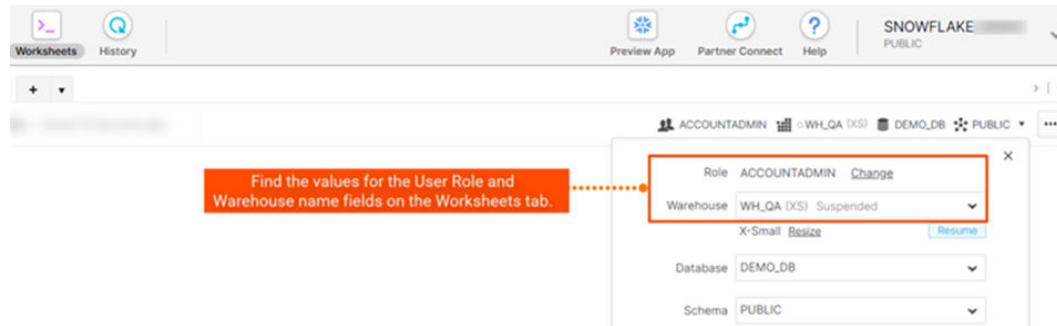
Authorization code

To connect to Snowflake using the OAuth 2.0 authorization code, you need the Snowflake client ID, authorization URL, access token URL, and access token.

To get the authorization details, you need to create an authorization integration in Snowflake, and register the Informatica redirect URL in Security Integration. Security Integration is a type of integration that enables clients that support OAuth to redirect users to an authorization page and generate access tokens, and optionally, refresh tokens to access Snowflake.

Register the following Informatica redirect URL in Security Integration:

```
https://<Informatica cloud hosting facility for your organization>/ma/proxy/oauthcallback
```

If the access token expires, Informatica redirect URL, which is outside the customer firewall, tries to connect to the endpoint and retrieves a new access token.

For more information about how to create a security integration and get the authorization details, see Create security integration in the Snowflake documentation.

Note: You can't use connections configured with the authorization code authentication in mappings configured in advanced mode.

Key pair

To connect to Snowflake using key pair authentication, you need the private key file and private key file password, along with your Snowflake account user name.

Generate the public and private key pair using OpenSSL. The key pair authentication method requires a 2048-bit RSA key pair. Specify the path to the private key file and password in the connection properties to access Snowflake.

Generate the public and private key

Before you generate the public and private key for key pair authentication, you need to have the security admin role or higher in Snowflake.

1. From the OpenSSL command line, generate a private key:
    - To generate a decrypted private key, run the following command, and provide a passphrase when prompted:
        ```
        $ openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -out rsa_key.p8 -nocrypt
        ```

- To generate an encrypted private key, run the following command, and provide a passphrase when prompted:

```
$ openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -out rsa_key.p8
```

The passphrase is used to encrypt the private key file while connecting to Snowflake.

2. Generate the public key. Run the following command and specify the encrypted private key located in the file, for example, `rsa_key.p8`:

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

3. Copy the public and private key files in a directory that the Secure Agent can access.
   For example, `C:\Program Files\Informatica Cloud Secure Agent\apps\Data_Integration_Server \data\snowflake\rsa_key.p8`

   You require the path details when you configure the Snowflake connection.

4. In Snowflake, assign the public key to the Snowflake user using the ALTER USER command:

```
alter user <user> set rsa_public_key='<content of the public key
after removing the header and footer lines>';
```

For example, `alter user jsmith set rsa_public_key='MIIXBIjABCdef...';`

For more information about configuring a key pair authentication for Snowflake, see the Snowflake documentation.

Configure the private key on an advanced cluster

After you generate the public and private key pair using OpenSSL, you need to additionally perform certain tasks for the connection to work in a mapping in advanced mode.

Before you run mappings with the configured connection on an advanced cluster, set the properties for the cluster application in the mapping task.

The following list describes the properties that you need to set in the advanced session properties in a mapping task:

Spark.NeedUserCredentialFileForAdapter=true

Copies the contents of the private key from the location you specify in `Spark.UserCredentialDirOnDIS` from the Secure Agent machine to the Spark driver and executers. The folder that contains the credential file does not have the 1 MB limit. You need to ensure that the credential file of the secret key content that you copy to the cluster application does not exceed 1 MB. You need to set the value to true. Default is false.

If you do not set this flag or you set this flag to false, the private key file is not copied to the cluster application and the mapping fails.

Spark.UserCredentialDirOnDIS=<private key file directory>

Overrides the default Secure Agent directory that contains the private key with the directory that you specify for copying the private key contents to the cluster application. The default directory is `/infa/ user/credentials`. Ensure that the directory does not include the private key file name.

If you do not set this flag, the default location is used. To use the default location, create the `/infa/ user/credentials` directory on the Secure Agent machine and the copy the private key file here.

If you set the flag to override the location specified in the advanced session properties of the mapping task, make sure that the override location that you specify in *Spark.UserCredentialDirOnDIS* contains the private key file. Ensure that the override location and the private key file have the write permissions.

The following image shows the configured advanced custom property in the mapping task:

Advanced Session Properties	
Session Property Name	**Session Property Value**
advanced.custom.property	Spark.NeedUserCredentialFileForAdapter=true&Spark.UserCredentialDirOnDIS=/cldagnt/pvtKey

Client credentials

To connect to Snowflake using OAuth 2.0 client credentials, you need your Snowflake client ID, access token URL, client secret, scope, and the access token.

Configure the OAuth endpoint with the client credentials grant type and then create a security integration to get the authorization details.

Before you use the client credentials authentication to connect Snowflake, the organization administrator needs to perform the prerequisite tasks.

1. Create a client application that is compatible with OAuth to use with Snowflake.

2. Configure the authorization server with the client credentials Grant type.

3. Create a security integration of type OAuth in Snowflake.
   For more information about how to create a security integration and get the authorization details, see Create security integration for external OAuth in the Snowflake documentation.

Note: You can't use connections configured with the client credentials authentication in mappings configured in advanced mode.

Connect to Snowflake

Let's configure the Snowflake Data Cloud connection properties to connect to Snowflake.

Before you begin

Before you get started, you'll need to get information from your Snowflake account based on the authentication type that you want to configure.

Check out "Prepare for authentication" on page 12 to learn more about the authentication prerequisites.

Connection details

The following table describes the basic connection properties:

Property	Description
Connection Name	Name of the connection. Each connection name must be unique within the organization. Connection names can contain alphanumeric characters, spaces, and the following special characters: _ . + -, Maximum length is 255 characters.
Description	Description of the connection. Maximum length is 4000 characters.

Property	Description
Type	Snowflake Data Cloud
Runtime Environment	The name of the runtime environment where you want to run tasks. Select a Secure Agent, Hosted Agent, or serverless runtime environment. For more information about how to configure and use the serverless environment, see "Serverless runtime environment setup" in *Runtime Environments* in the Administrator help. For more information about how to configure a serverless environment with key pair authentication, see "Use the serverless runtime environment with key pair authentication" on page 25. Do not use a Hosted Agent if you use the connection in mappings in advanced mode.

Authentication types

You can configure standard, authorization code, key pair, and client credentials authentication types to access Snowflake.

Select the required authentication method and then configure the authentication-specific parameters.

Standard authentication

Standard authentication is the default type which requires at a minimum your Snowflake account name and password.

The following table describes the basic connection properties for standard authentication:

Property	Description
Username	The user name to connect to the Snowflake account.
Password	The password to connect to the Snowflake account.
Account	The name of the Snowflake account. For example, if the Snowflake URL is https://<123abc>.us-east-2.aws.snowflakecomputing.com/console/login#/, your account name is the first segment in the URL before snowflakecomputing.com. Here, 123abc.us-east-2.aws is your account name. If you use the Snowsight URL, for example, https://app.snowflake.com/us-east-2.aws/<123abc>/dashboard, your account name is 123abc.us-east-2.aws. **Note:** Ensure that the account name doesn't contain underscores. If the account name contains underscores, you need to use the alias name. To use an alias name, contact Snowflake Customer Support.
Warehouse	The Snowflake warehouse name.

Advanced settings

The following table describes the advanced connection properties for standard authentication:

Property	Description
Role	The Snowflake role assigned to the user.

Additional JDBC URL Parameters	The additional JDBC connection parameters.
	
	You can specify multiple JDBC connection parameters, separated by ampersand (&), in the following format:
	
	`<param1>=<value>&<param2>=<value>&<param3>=<value>....`
	
	For example, you can pass the following database and schema values when you connect to Snowflake:
	
	`db=mydb&schema=public`
	
	When you add parameters, ensure that there is no space before and after the equal sign (=).
	
	For the list of additional JDBC parameters that you can configure, see "JDBC URL parameters" on page 23.

Authorization code authentication

OAuth 2.0 authentication requires the OAuth 2.0 protocol with Authorization Code grant type to connect to Snowflake. Authorization Code allows authorized access to Snowflake without the need to share or store your login credentials.

The following table describes the basic connection properties for OAuth 2.0 authorization code authentication:

Property	Description
Account	The name of the Snowflake account. For example, if the Snowflake URL is https://<123abc>.us-east-2.aws.snowflakecomputing.com/console/login#/, your account name is the first segment in the URL before snowflakecomputing.com. Here, 123abc.us-east-2.aws is your account name. If you use the Snowsight URL, for example, https://app.snowflake.com/us-east-2.aws/<123abc>/dashboard, your account name is 123abc.us-east-2.aws. **Note:** Ensure that the account name doesn't contain underscores. If the account name contains underscores, you need to use the alias name. To use an alias name, contact Snowflake Customer Support.
Warehouse	The Snowflake warehouse name.

Property	Description
Authorization URL	The Snowflake server endpoint that is used to authorize the user request.
	The authorization URL is https://<account name>.snowflakecomputing.com/oauth/authorize, where <account name> specifies the full name of your account provided by Snowflake.
	For example, https://<abc>.snowflakecomputing.com/oauth/authorize
	Note: If the account name contains underscores, use the alias name.
	You can also use the Authorization Code grant type that supports the authorization server in a Virtual Private Cloud network.
Access Token URL	The Snowflake access token endpoint that is used to exchange the authorization code to get an access token.
	The access token URL is https://<account name>.snowflakecomputing.com/oauth/token-request, where <account name> specifies the full name of your account provided by Snowflake.
	For example, https://<abc>.snowflakecomputing.com/oauth/token-request
	Note: Ensure that the account name doesn't contain underscores. If the account name contains underscores, you need to use the alias name. To use an alias name, contact Snowflake Customer Support.
Client ID	Client ID of your application generated when you create a security integration of type OAuth in Snowflake.
Client Secret	Client secret generated for the client ID.
Access Token	The access token value.
	Enter the populated access token value that you get from the OAuth endpoint, or click **Generate Access Token** to populate the access token value.

Advanced settings

The following table describes the advanced connection properties for OAuth 2.0 authorization code authentication:

Property	Description

Additional JDBC URL Parameters	The additional JDBC connection parameters.
	
	You can specify multiple JDBC connection parameters, separated by ampersand (&), in the following format:
	
	`<param1>=<value>&<param2>=<value>&<param3>=<value>....`
	
	For example, you can pass the following database and schema values when you connect to Snowflake:
	
	`db=mydb&schema=public`
	
	When you add parameters, ensure that there is no space before and after the equal sign (=).
	For the list of additional JDBC parameters that you can configure, see "JDBC URL parameters" on page 23.

Scope	Determines the access control when the API endpoint has defined custom scopes. For example, specify `session:role:CQA_GCP` as the scope to override the value of the default user role. The value needs to be one of the roles assigned in Security Integration. To enter multiple scope attributes, separate each scope attribute with a space.

Access Token Parameters	Additional parameters to use with the access token URL.
	
	Define the access token parameters in the following JSON format:
	
	`[{"Name":"<Parameter name>","Value":"<Parameter value>"}]`
	
	For example, you can use the following code_verifier parameter when you connect to Snowflake:
	
	`[{"Name":"code_verifier","Value":"5PMddu6Zcg6Tc4sbg"}]`
	
	For more information about access token parameters that you can define, see Introduction to OAuth in the Snowflake documentation.

Property	Description
Authorization Code Parameters	Additional parameters to use with the authorization token URL.
	Define multiple parameters, separated by comma, in the following JSON format:
	`[{"Name":"<Parameter name>","Value":"<Parameter value>"}, {"Name":"<Parameter name>","Value":"<Parameter value>"}]`
	For example, you can use the following code_challenge and code_challenge_method parameters when you connect to Snowflake:
	`[{"Name":"code_challenge","Value":"Ikr-vv52th0UeVRi4"}, {"Name":"code_challenge_method","Value":"S256"}]`
Refresh Token	The refresh token value.
	Enter the populated refresh token value that you get from the OAuth endpoint, or click **Generate AccessToken** to populate the refresh token value. If the access token is not valid or expires, the Secure Agent fetches a new access token with the help of the refresh token.
	Note: If the refresh token expires, provide a valid refresh token or regenerate a new refresh token by clicking **Generate Access Token**.

Key pair authentication

Key pair authentication requires the private key file and private key file password, along with your Snowflake account user name to connect to Snowflake.

The following table describes the basic connection properties for key pair authentication:

Property	Description
Username	The user name to connect to the Snowflake account.
Account	The name of the Snowflake account.
	For example, if the Snowflake URL is https://<123abc>.us-east-2.aws.snowflakecomputing.com/console/login#/, your account name is the first segment in the URL before snowflakecomputing.com. Here, 123abc.us-east-2.aws is your account name.
	If you use the Snowsight URL, for example, https://app.snowflake.com/us-east-2.aws/<123abc>/dashboard, your account name is 123abc.us-east-2.aws.
	Note: Ensure that the account name doesn't contain underscores. If the account name contains underscores, you need to use the alias name. To use an alias name, contact Snowflake Customer Support.

Property	Description
Warehouse	The Snowflake warehouse name.
Private Key File	Path to the private key file, including the private key file name, that the Secure Agent uses to access Snowflake. For example, specify the following path and key file name in the Secure Agent machine: - **On Windows:** `C:\Users\path_to_key_file\rsa_key.p8` - **On Linux:** `/export/home/user/path_to_key_file/rsa_key.p8` To use the serverless runtime environment, specify the following path and key file name in the serverless agent directory: `/home/cldagnt/SystemAgent/serverless/configurations/ssl_store/<Private key file name>` For more information about how to use the serverless environment, see "Use the serverless runtime environment with key pair authentication" on page 25. **Note:** Verify that the keystore is FIPS-certified.

Advanced settings

The following table describes the advanced connection properties for key pair authentication:

Property	Description

Additional JDBC URL Parameters	The additional JDBC connection parameters.
	
	You can specify multiple JDBC connection parameters, separated by ampersand (&), in the following format:
	`<param1>=<value>&<param2>=<value>&<param3>=<value>....`
	
	For example, you can pass the following database and schema values when you connect to Snowflake:
	`db=mydb&schema=public`
	
	When you add parameters, ensure that there is no space before and after the equal sign (=).
	For the list of additional JDBC parameters that you can configure, see "JDBC URL parameters" on page 23.

Private Key File Password	Password for the private key file.

Client credentials authentication

OAuth 2.0 client credentials authentication requires at a minimum the client ID, access token URL, client secret, scope, and the access token.

The following table describes the basic connection properties for OAuth 2.0 client credentials authentication:

Property	Description
Account	The name of the Snowflake account. For example, if the Snowflake URL is https://<123abc>.us-east-2.aws.snowflakecomputing.com/console/login#/, your account name is the first segment in the URL before snowflakecomputing.com. Here, 123abc.us-east-2.aws is your account name. If you use the Snowsight URL, for example, https://app.snowflake.com/us-east-2.aws/<123abc>/dashboard, your account name is 123abc.us-east-2.aws. **Note:** Ensure that the account name doesn't contain underscores. If the account name contains underscores, you need to use the alias name. To use an alias name, contact Snowflake Customer Support.
Warehouse	The Snowflake warehouse name.
Access Token URL	The Snowflake access token endpoint that is used to exchange the authorization code for an access token. Specify the access token URL that you get from the OAuth endpoint.
Client ID	Client ID of your application generated when you configure the application for OAuth.
Client Secret	Client secret generated for the client ID.
Scope	Determines the access control when the API endpoint has defined custom scopes. For example, specify `session:role:CQA_GCP` as the scope to override the value of the default user role. The value needs to one of the roles assigned in Security Integration. To enter multiple scope attributes, separate each scope attribute with a space.
Access Token	The access token value. Enter the populated access token value that you get from the OAuth endpoint, or click **Generate Access Token** to populate the access token value.

Advanced settings

The following table describes the advanced connection properties for OAuth 2.0 client credentials authentication:

Property	Description

Additional JDBC URL Parameters	The additional JDBC connection parameters.
	
	You can specify multiple JDBC connection parameters, separated by ampersand (&), in the following format::
	
	`<param1>=<value>&<param2>=<value>&<param3>=<value>....`
	
	For example, you can pass the following database and schema values when you connect to Snowflake:
	
	`db=mydb&schema=public`
	
	When you add parameters, ensure that there is no space before and after the equal sign (=).
	For the list of additional JDBC parameters that you can configure, see "JDBC URL parameters" on page 23.

Access Token Parameters	Additional parameters to use with the access token URL.
	
	Define the access token parameters in the following JSON format:
	
	`[{"Name":"<Parameter name>","Value":"<Parameter value>"}]`
	
	For example, you can use the following code_verifier parameter when you connect to Snowflake:
	
	`[{"Name":"code_verifier","Value":"5PMddu6Zcg6Tc4sbg"}]`
	
	For more information about access token parameters that you can define, see Introduction to OAuth in the Snowflake documentation.

JDBC URL parameters

You can use the additional JDBC URL parameters field in the Snowflake Data Cloud connection to customize and set any additional parameters when you connect to Snowflake.

You can configure the following properties as additional JDBC URL parameters in the Snowflake Data Cloud connection:

- To override the database and schema name used to create temporary tables in Snowflake, enter the database and schema name in the following format:

      `ProcessConnDB=<DB name>&ProcessConnSchema=<schema_name>`

- To view only the specified database and schema while importing a Snowflake table, enter the database and schema name in the following format:

      `db=<database_name>&schema=<schema_name>`

- To read UDF string and numeric data from Snowflake, enter the database and schema where the UDF is created in Snowflake in the following format:

      `db=<database_name>&schema=<schema_name>`

- To access Snowflake through Okta SSO authentication, enter the web-based IdP implementing SAML 2.0 protocol in the following format:

  ```
  authenticator=https://<Your_Okta_Account_Name>.okta.com
  ```

  Note: Microsoft ADFS is not applicable.

  For more information about configuring Okta authentication, see
  [Configuring Snowflake to use federated authentication](#).

- To load data from Google Cloud Storage or Microsoft Azure Data Lake Storage Gen2 to Snowflake for SQL ELT optimization, enter the Cloud Storage Integration name created for the Google Cloud Storage or Microsoft Azure Data Lake Storage Gen2 account in Snowflake in the following format:

  ```
  storage_integration=<Storage Integration name>
  ```

  The storage integration name is case-sensitive. For example, if the storage integration name you created for Google Cloud Storage or Microsoft Azure Data Lake Storage Gen2 in Snowflake is *STORAGE_INT*, you need to specify the same integration name:

  ```
  storage_integration=STORAGE_INT
  ```

- To connect to Snowflake using the proxy server, enter the following parameters:

  ```
  useProxy=true&
  proxyHost=<Proxy host IP address>&
  proxyPort=<Proxy server port number>&
  proxyUser=<Proxy server user name>&
  proxyPassword=<Proxy server password>
  ```

- To ignore double quotes in the table and treat all tables as case-insensitive, enter the following parameter:

  ```
  QUOTED_IDENTIFIERS_IGNORE_CASE=true
  ```

  When you set this property in the connection to true, Snowflake ignores the double quotes in the table and treats all tables as case-insensitive.
  If you have set this property to true, you cannot access case-sensitive tables with the same connection. You need to create a new connection to fetch any existing case-sensitive tables.

- To filter queries that are executed in a Snowflake job on the Snowflake web interface, enter the tag name in the following format:

  ```
  query_tag=<Tag name>
  ```

  You have an option to override the query_tag parameter that is defined in the Snowflake connection when you run a mapping task.

  To override the query_tag parameter, click the **Runtime Options** tab of the mapping task. In the **Advanced Session Properties** section, select **Custom Properties** from the **Session Property Name** list, and then enter the following value:

  ```
  snowflake_query_tag=<Tag name>
  ```

  Note: In advanced mode, you can't override the query_tag parameter.

In addition to the parameters listed, this field provides you the flexibility to configure other Snowflake parameters based on your requirements.

Proxy server settings

If your organization uses an outgoing proxy server to connect to the Internet, the Secure Agent connects to Informatica Intelligent Cloud Services through the proxy server.

You can configure the Secure Agent to use the proxy server on Windows and Linux. You can use the unauthenticated or authenticated proxy server. The proxy settings applies to connections used in mappings and in mappings in advanced mode.

To configure proxy settings for the Secure Agent, use one of the following methods:

- Configure the Secure Agent through the Secure Agent Manager on Windows or shell command on Linux.

  For instructions, see "Configure the proxy settings on Windows" or "Configure the proxy settings on Linux" in *Getting Started* in the Data Integration help.

- Configure the JVM options for the DTM in the Secure Agent properties. For instructions, see the Proxy server settings Knowledge Base article.

- Configure the proxy server properties in the additional JDBC URL parameters in the Snowflake connection. For more information, see Set JDBC URL Parameters.

To configure proxy settings for the serverless runtime environment, see "Using a proxy server" in *Runtime Environments* in the Administrator help.

Private links to access Snowflake

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

When you create a Snowflake Data Cloud connection, specify the Snowflake private link account name in the **Account** field in the connection properties.

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.

Use the serverless runtime environment with key pair authentication

You can use a serverless runtime environment hosted on AWS or Azure to connect to Snowflake with key pair authentication.

Before you configure a Snowflake connection using the serverless runtime environment, perform the following tasks:

- Add the private key file path and file name in the Amazon S3 bucket or Azure container in your AWS or Azure account.
- Configure the .yml serverless configuration file.
- Configure the connection properties to connect to Snowflake.

Add the private key file path and file name in the Amazon S3 bucket or Azure container in your AWS or Azure account

Perform the following steps to configure a Snowflake connection in a serverless runtime environment:

1. Create the following structure for the serverless agent configuration in AWS or Azure:
   ```
   <Supplementary file location>/serverless_agent_config
   ```

2. Add the path to the private key file, including the private key file name, in the Amazon S3 bucket or Azure container in the following location in your AWS or Azure account: `<Supplementary file location>/serverless_agent_config/SSL`

Configure the .yml serverless configuration file

Perform the following steps to configure the .yml serverless configuration file in the serverless runtime environment, and to copy the private key file path and file name entries to the serverless agent directory:

1. Copy the following code snippet to a text editor:

```
version: 1
agent:
  agentAutoApply:
    general:
      sslStore:
        - fileCopy:
            sourcePath: SSL/<Private key file name>
```

where the source path is the directory path of the private key file in AWS or Azure.

2. Ensure that the syntax and indentations are valid, and then save the file as `serverlessUserAgentConfig.yml` in the following AWS or Azure location: `<Supplementary file location>/serverless_agent_config`

When the .yml file runs, the private key file is copied from the AWS or Azure location to the serverless agent directory.

Configure the connection properties to connect to Snowflake

Specify the path to the private key file, including the private key file name in the **Private Key File** field in the Snowflake Data Cloud connection.

For example, `/home/cldagnt/SystemAgent/serverless/configurations/ssl_store/<Private key file name>`

For more information about how to configure and use the serverless environment, see "Serverless runtime environment setup" in *Runtime Environments* in the Administrator help.

Part II: Data Integration with Snowflake Data Cloud Connector

This part contains the following chapters:

CHAPTER 3

Mappings for Snowflake Data Cloud

When you configure a mapping, you describe the flow of data from the source to the target. A mapping defines reusable data flow logic that you can use in mapping tasks.

When you create a mapping, you define the Source, Target, and Lookup transformations to represent a Snowflake Data Cloud object. Use the Mapping Designer in Data Integration to add the Source, Target, or Lookup transformations in the mapping canvas and configure the source, target, and lookup properties. To create a mapping, select **Mapping** as the mapping type in the Mapping Designer.

This chapter provides instructions on configuring mappings and mappings in advanced mode for Snowflake Data Cloud. You can add mappings and mappings in advanced mode to the mapping task and enable SQL ELT optimization in the mapping task. For more information, see Chapter 11, "SQL ELT optimization for mapping tasks" on page 89.

You can also create a mapping in SQL ELT mode when your source and target are in the same cloud ecosystem and you want to perform the data transformation entirely within the cloud ecosystem. For more information about mappings in SQL ELT mode, see Chapter 10, "Mappings in SQL ELT mode for Snowflake Data Cloud" on page 76.

Before you begin

Before you begin configuring and running a mapping, complete the required prerequisites.

Verify permissions

Permissions define the level of access for the operations that you can perform in Snowflake.

The following table lists the permissions that you require in the Snowflake account:

Object Type	Privileges
Database	Usage
Schema	Usage, Create Table, Create View, Create Sequence
Table	All

Object Type	Privileges
Sequence	All
Stored Procedure	All

For more information, see Access Control Privileges in the Snowflake documentation.

Verify ports

Ensure that ports 443 and 80 in Snowflake are open for access.

Snowflake Data Cloud transformations

Add transformations to the mapping to represent the operations that you want to perform on data.

You can add the following transformations and define the transformation details in a mapping:

Source transformation

Add a Source transformation to read data from Snowflake or other data sources. You can read from a single or from multiple Snowflake tables. You can also use a query or parameter as the source type to read from Snowflake.

You can use one or more Source transformations in a mapping. When you use a single transformation to read from multiple tables, you can use a join. If you use two Source transformations in a mapping, use a Joiner transformation to join the data.

Target transformation

Add a Target transformation to write data to Snowflake. You can use one or more Target transformations in a mapping. When you configure a target transformation, you an use a single Snowflake target object. You can use a new or an existing Snowflake target object. You can also use parameters to define the target connection and target object when you run the mapping task.

Lookup transformation

Add a Lookup transformation to retrieve data based on a specified lookup condition. The lookup object can be a single object or query. You can also parameterize the lookup objects and connections.

For more information about the objects and the properties that you can configure for each of these transformations in a mapping or in a mapping in advanced mode, see the Chapter 4, "Sources for Snowflake Data Cloud" on page 38, Chapter 5, "Targets for Snowflake Data Cloud" on page 44, and Chapter 6, "Lookups for Snowflake Data Cloud " on page 60 chapters.

You can also add other transformations to the mapping to transform the data. For more information about configuring transformations, see *Transformations* in the Data Integration documentation.

SQL transformation

You can configure an SQL transformation in a Snowflake Data Cloud mapping to process SQL queries and stored procedures in Snowflake.

When you add an SQL transformation to the mapping, on the **SQL** tab, you define the database connection and the type of SQL that the transformation processes.

You can choose to use a parameterized connection in a SQL transformation. You can also override the values defined in a parameter file at runtime. To use a parameterized connection in an SQL transformation, first create an SQL transformation in a mapping that uses a valid connection. Then, parameterize the connection in the SQL transformation. You can also use an SQL transformation to read from Java or SQL user-defined functions (UDF) in Snowflake.

You can use an SQL transformation to process the following types of SQL statements:

Stored procedure

You can configure an SQL transformation to call stored procedures and overloaded stored procedures in Snowflake. The stored procedure name is case-sensitive. You can select the stored procedure from the database, or enter the exact name of the stored procedure to call in the SQL transformation. Before you create the SQL transformation, ensure that stored procedure exists in the Snowflake database.

When you specify the Snowflake database, schema, and procedure name in the advanced SQL properties, the agent considers the properties specified in the stored procedure first, followed by the advanced source properties, then the additional JDBC URL parameters in the connection, and finally the source object metadata.

If you add a new stored procedure to the database when the mapping is open, the new stored procedure does not appear in the list of available stored procedures. To refresh the list, close and reopen the mapping.

SQL Query

You can configure an SQL transformation to process an entered query that you define in the SQL editor. Do not use more than one SQL query in an SQL transformation.

The SQL transformation processes the query and returns the rows. The SQL transformation also returns any errors that occur from the underlying database or if there is an error in the user syntax.

For more information about SQL queries and stored procedures, see *Transformations* in the Data Integration documentation.

Rules and guidelines

You can use a SQL transformation with the following restrictions:

- You cannot use the saved query type in an SQL transformation.
- Mappings fail when you specify schema override results in multiple stored procedures that contain the same name and number of arguments.
- Mappings fail when the stored procedure contains Unicode characters.
- If NULL is returned from a stored procedure, a warning appears in the user interface and the session log. However, the mapping continues to process the rows.
- The runtime processing ignores the following properties in an SQL transformation:
    - In-out and input parameters
    - Advanced properties
    - Auto commit

- Transformation scope

- Stop on error

• You cannot use key range partitioning when the mapping contains an SQL transformation.

Handling dynamic schemas

When you add a mapping to a mapping task, you can choose how Data Integration handles changes in the data object schemas. To refresh the schema every time the task runs, you can enable dynamic schema handling in the task.

A schema change includes one or more of the following changes to the data object:

• Fields added, deleted, or renamed.

• Fields updated for data type, precision, or scale.

Configure schema change handling in the **Advanced Options** section on the **Runtime Options** tab when you configure the task. You can configure asynchronous or dynamic schema change handling.

When you configure dynamic schema change handling, you can choose from the following options to refresh the schema:

Alter and apply changes

Data Integration applies the following changes from the source schema to the target schema:

• New fields. Alters the target schema and adds the new fields from the source.

• Renamed fields. Adds renamed fields as new columns in the target.

• Data type and precision updates. Applies these changes to the target. Updates to the scale are not applicable.

• Deleted fields. Ignores deleted fields.

Note: To include schema changes for renamed or new fields for mappings in advanced mode, see .

Don't apply DDL changes

Data Integration does not apply the schema changes to the target.

Drop current and recreate

Drops the existing target table and then recreates the target table at runtime using all the incoming metadata fields from the source.

Handling dynamic schemas for mappings in advanced mode

When you run a mapping in advanced mode and if a column is renamed or a field is added to the source, the target schema is jumbled because of the difference in columns in the source and target schema.

To avoid this, set the following properties, separated by an ampersand, in the **Additional Write Runtime Parameters** field in the Target transformation:

• column_mapping=name

• column_mismatch_behavior=ignore

When you set these properties, Data Integration includes the name and also ignores any mismatch between the source and target schema when the column names contain uppercase letters, digits, or underscore.

If the column names do not contain uppercase letters, you must additionally set the keep_column_case=on parameter in both the Source and Target transformations:

- For the source, set the parameter in the **Additional JDBC URL Parameters** of the Snowflake Data Cloud connection.
- For the target, specify the parameter in the **Additional Write Runtime Parameters** property of the Target transformation.

Rules and guidelines for dynamic schema handling

Consider the following rules and guidelines when you enable dynamic schema change handling:

- Do not include an override for the target or source from the advanced properties. If the mapping is enabled for dynamic schema handling and contains a source or target override, the mapping fails with the following error: `Exception: class java.lang.NullPointerException occurred in writeMetadata` However, you can specify an SQL override in mappings enabled for dynamic schema handling.
- When you set the keep_column_case=on parameter in the connection properties to fix dynamic schema changes for column names that are not in upper case letters, ensure to use this connection only for mappings in advanced mode. Data Integration uses this connection to run the mapping on an advanced cluster. If you use the same connection in mappings that are not in advanced mode, it results in an error.
- In advanced mode, do not specify the **Truncate Target Table** option in the Target transformation properties. If a column is renamed or added in the source schema and if you configure the **Truncate Target Table** and also set the **Additional Write Runtime Parameters** for the Target transformation, the mapping fails with the following error: `java.lang.UnsupportedOperationException`

Mapping example

An enterprise application uses the Oracle database to store product transaction details. You want to check the availability of stocks based on completed and pending transactions from the source data. You want to integrate the available stocks and transaction details to Snowflake for further analysis.

Mapping

The following image shows the Snowflake Data Cloud mapping for this use case:
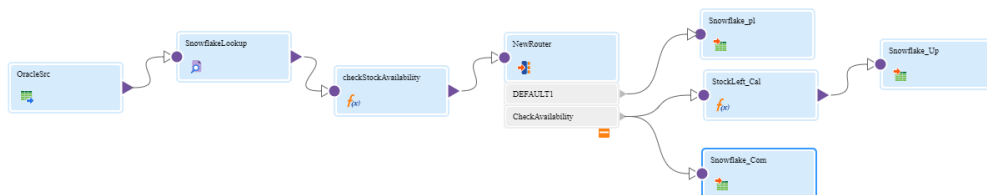
You create a mapping to read the product transaction details from an Oracle source and apply a lookup condition on the PRODUCTDET table in Snowflake which stores details of product and its availability. Based on the availability and requirement, you write the transactions to the PENDINGTRANSACTION and COMPLETEDTRANSACTION tables in Snowflake and update the INSTOCK field in the PRODUCTDET table for the completed transactions.

You use the following transformations in the Snowflake Data Cloud mapping:

Source transformation

To read the product transaction details from an Oracle source, include an Oracle connection in the Source transformation to connect to Oracle. The source object for the mapping task is an OracleSrc table from Oracle.

The following image shows the transaction details stored in the OracleSrc table that you want to read:

transactionID	CustomerID	productID	quantity	OrderPlacedOn
Tran511	CUST21	P45	100	2016-04-05
Tran512	CUST22	P46	200	2016-07-05
Tran513	CUST23	P47	20	2016-07-25
Tran514	CUST24	P47	100	2016-10-25
Tran515	CUST25	P45	1000	2016-12-02
Tran517	CUST27	P46	5000	2017-01-02
Tran516	CUST26	P48	60	2017-01-02
Tran518	CUST28	P49	60	2017-01-03
Tran519	CUST29	P50	700	2017-03-13
Tran520	CUST30	P47	750	2017-03-14

Lookup transformation

The lookup object for the mapping task is PRODUCTDET table in Snowflake, which has details of the product and its availability. Apply the lookup condition on the PRODUCTDET table in Snowflake which stores details of product and its availability based on the product ID.

The following image shows the data stored in the PRODUCTDET table:

Data Preview

Connection: snowflake_CQA   Object: PRODUCTDET

PRODUCTID	INSTOCK	PRODUCTDET	PRICE
p45	900	2.5" 80GB IDE Laptop Har...	1968
p46	10000	Laptop Internal CD/DVD R...	1229
p47	5000	New HP ProBook 430 G3 ...	5289
p48	50	New HP ProBook 430 G3 ...	9594
p49	20	Dell Inspiron 15R N5110 B...	1699
p50	800	HP 15-be016TU 15.6-inch...	27490

Expression transformation

The Expression transformation uses an expression to calculate the stock availability.

Router transformation

The Router transformation filters data based on the availability of stocks and redirects completed transactions, pending transactions, and product details to the appropriate target tables.

Target transformation

The mapping task has the following target objects to write the completed transactions, pending transactions, and product details:

COMPLETEDTRANSACTION

The COMPLETEDTRANSACTION table includes the TRANSACTIONID, PRODUCTID, QUANTITY, ORDERPLACEDON, and ORDERCOMPLETEDON fields.

The following image shows the data stored in the COMPLETEDTRANSACTION table:

Connection: snowflake_CQA		Object: COMPLETEDTRANSACTION		
TRANSACTIONID	PRODUCTID	QUANTITY	ORDERPLACEDON	ORDERCOMPLETEDO
Tran511	P45	100	2016-04-05 00:00:00.0	2016-04-05 00:00:00.0
Tran512	P46	200	2016-07-05 00:00:00.0	2016-07-05 00:00:00.0
Tran513	P47	20	2016-07-25 00:00:00.0	2016-07-25 00:00:00.0
Tran514	P47	100	2016-10-25 00:00:00.0	2016-10-25 00:00:00.0
Tran517	P46	5000	2017-01-02 00:00:00.0	2017-01-02 00:00:00.0
Tran519	P50	700	2017-03-13 00:00:00.0	2017-03-13 00:00:00.0
Tran520	P47	750	2017-03-14 00:00:00.0	2017-03-14 00:00:00.0

PENDINGTRANSACTION

The PENDINGTRANSACTION table includes the PRODUCTID, TRANSACTIONID, REQUIREDQUANTITY, and ORDERPLACEDON fields.

The following image shows the data stored in the PENDINGTRANSACTION table:

Connection: snowflake_CQA		Object: PENDINGTRANSACTION	
PRODUCTID	TRANSACTIONID	REQUIREDQUANTITY	ORDERPLACEDON
P45	Tran515	1000	2016-12-02 00:00:00.0
P48	Tran516	60	2017-01-02 00:00:00.0
P49	Tran518	60	2017-01-03 00:00:00.0

PRODUCTDET

The PRODUCTDET table includes the PRODUCTID, INSTOCK, PRODUCTDET, and PRICE fields. Based on the completed transactions, the INSTOCK field is updated.

The following image shows the data stored in the PRODUCTDET table:

Connection: snowflake_CQA		Object: PRODUCTDET	
PRODUCTID	INSTOCK	PRODUCTDET	PRICE
P48	50	New HP ProBook 430 G3 ...	9594
P49	20	Dell Inspiron 15R N5110 B...	1699
P45	800	2.5" 80GB IDE Laptop Har...	1968
P46	4800	Laptop Internal CD/DVD R...	1229
P47	4130	New HP ProBook 430 G3 ...	5289
P50	100	HP 15-be016TU 15.6-inch...	27490

When you run the mapping, the agent reads the transaction details from source, fetches fields from the lookup, and based on the conditions applied write the available quantity and transaction details to the target tables.

Mappings in advanced mode example

You work for a retail company that offers more than 50,000 products and the stores are distributed across the globe. The company ingests a large amount of customer engagement details from the transactional CRM system into Amazon S3.

The sales team wants to improve customer engagement and satisfaction at every touch point. To create a seamless customer experience and deliver personalized service across the various outlets, the retail

company plans to load the data that is stored in the parquet file format from the Amazon S3 bucket to Snowflake.

You can create a mapping in advanced mode to read data from the Amazon S3 bucket and write data to the Snowflake target. You can choose to add transformations in the mapping to process the raw data that you read from the Amazon S3 bucket and then write the curated data to Snowflake.

The following example illustrates how to create a mapping to read from an Amazon S3 source and write to Snowflake:

1. In Data Integration, click **New** > **Mappings** > **Mapping**.
2. In the Mapping Designer, click **Switch to Advanced**.

    The following image shows the **Switch to Advanced** button in the Mapping Designer:
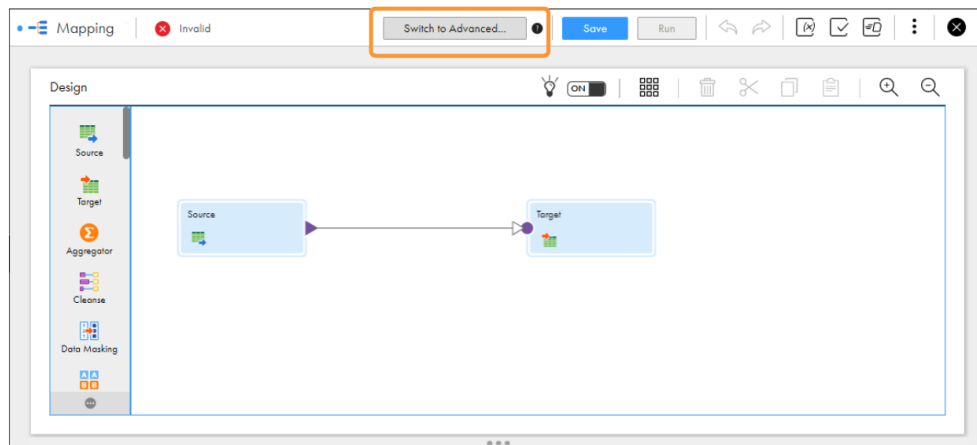
3. In the **Switch to Advanced** dialog box, click **Switch to Advanced**.

    The Mapping Designer updates the mapping canvas to display the transformations and functions that are available in advanced mode.

4. Enter a name, location, and description for the mapping.
5. Add a Source transformation, and specify a name and description in the general properties.
6. On the **Source** tab, perform the following steps to read data from the Amazon S3 source:

    a. In the **Connection** field, select the Amazon S3 V2 connection.

    b. In the **Source Type** field, select single object as the source type.

    c. In the **Object** field, select the parquet file object that contains the customer details.

    d. In the **Advanced Properties** section, specify the required parameters.

The following image shows the configured Source transformation properties that reads customer engagement details from the Amazon S3 object:
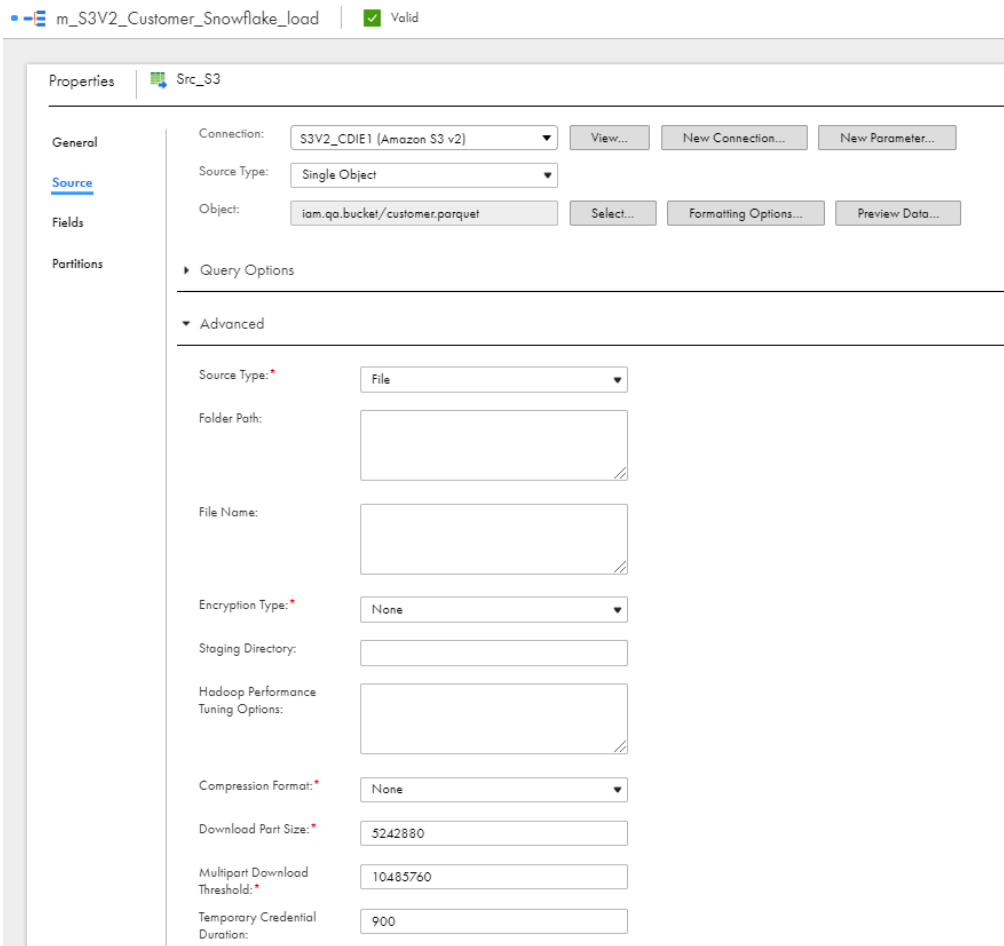
7. On the **Expression** tab, define an expression to change the file name port of the customer parquet file to uppercase based on your business requirement before you write data to the Snowflake target:

The following image shows the configured Expression transformation properties:
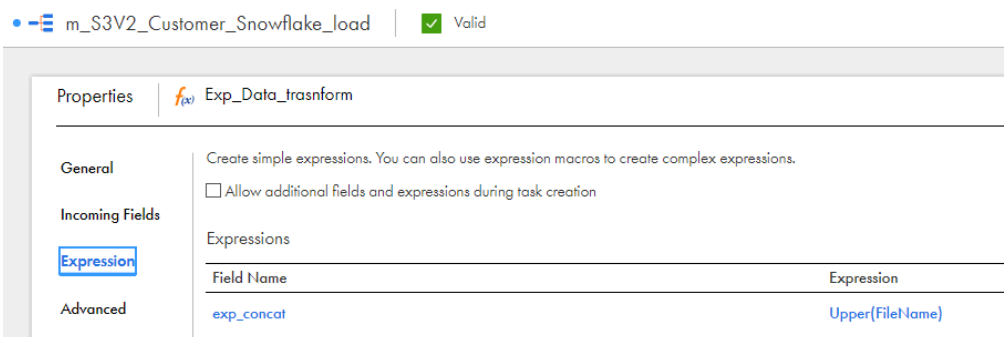
8. Add a Target transformation, and specify a name and description in the general properties.

9. On the **Target** tab, specify the details to write data to Snowflake:

   a. In the **Connection** field, select the Snowflake Data Cloud target connection.

b. In the **Target Type** field, select single object.

c. In the **Object** field, select the Snowflake object to which you want to write the curated customer engagement data.

d. In the **Operation** field, select the insert operation.

e. In the **Advanced Properties** section, specify the required advanced target properties.

The following image shows the configured Snowflake Target transformation properties:
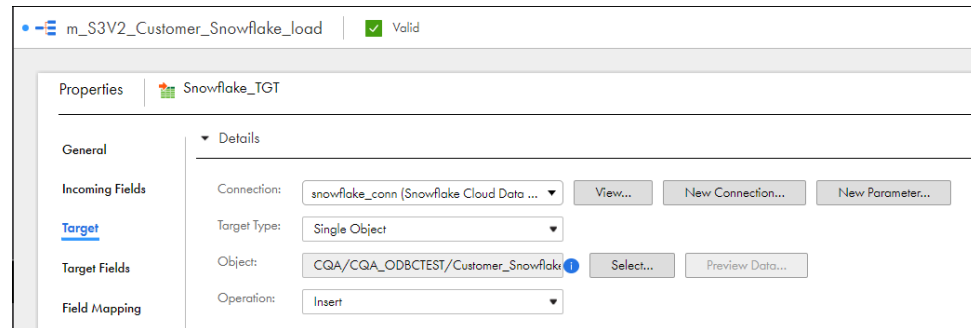
10. Click **Save** > **Run** to validate the mapping.

In Monitor, you can monitor the status of the logs after you run the task.

Parameterization restrictions for mappings in advanced mode

Consider the following restrictions for mappings in advanced mode:

- You cannot parameterize only the connection in a mapping. You need to parameterize both the connection and the object for the mapping to run successfully.

- Select the field mapping as Parameterized or Automatic. Manual field mapping does not work when you parameterize the source and target connection and objects.

- When you parameterize an object and specify an override, the metadata of the overridden object needs to match the specified object in the task properties.

- You cannot parameterize the object when you create a new target at runtime.

- When you override a parameter using the parameter file, you cannot use input parameters for the source filter, custom query, multiple objects, and the source, lookup, and target advanced source properties.

- You can override using the parameter file only if you have defined a parameter in the mapping.

CHAPTER 4

Sources for Snowflake Data Cloud

Add a Source transformation to extract data from a source. When you add a Source transformation to a mapping, you define the source connection, source objects, and source properties related to the Snowflake Data Cloud connection type.

Source properties for Snowflake Data Cloud

In a mapping, you can configure a Source transformation to represent a Snowflake Data Cloud source.

The following table describes the Snowflake Data Cloud source properties that you can configure in a Source transformation:

Property	Description
Connection	Name of the source connection. You can select an existing connection, create a new connection, or define parameter values for the source connection property. **Note:** You can switch between a non-parameterized and a parameterized Snowflake Data Cloud connection. The advanced property values are retained during the switch. If you want to overwrite the source connection properties at runtime, select the **Allow parameter to be overridden at run time** option. Specify the parameter file directory and name in the advanced session properties.
Source Type	Type of the source object. Select Single Object, Multiple Objects, Query, or Parameter.
Parameter	A parameter file where you define values that you want to update without having to edit the task. Select an existing parameter for the source object, or click **New Parameter** to define a new parameter for the source object. The **Parameter** property appears only if you select parameter as the source type. If you want to overwrite the parameter at runtime, select the **Allow parameter to be overridden at run time** option when you create a parameter. When the task runs, it uses the parameters from the file that you specify in the task advanced session properties.
Object	The source object for the task. Select the source object for a single source. When you select the multiple source option, you can add multiple source objects and configure a relationship between them.
Filter	Filters records based on the filter condition. You can specify a simple filter or an advanced filter.

Property	Description
Sort	Sorts records based on the conditions you specify. You can specify the following sort conditions: - Not parameterized. Select the fields and type of sorting to use. - Parameterized. Use a parameter to specify the sort option.
Select Distinct Rows Only[1]	Extracts distinct rows from the source table. **Note:** If you select the source type as query or you specify the SQL query property, the Select Distinct option is ignored
[1]Doesn't apply to mappings in advanced mode.	

The following table describes the advanced properties that you can configure in a Source transformation:

Advanced Property	Description
Database	Overrides the database specified in the connection.
Schema	Overrides the schema specified in the connection.
Warehouse	Overrides the Snowflake warehouse name specified in the connection.
Role	Overrides the Snowflake role assigned to user you specified in the connection. The warehouse name in the mapping overrides the warehouse name you specify in the connection. Even though you provide an incorrect warehouse name in the connection properties, the connection is successful. However, before you run the mapping, ensure that you specify the correct warehouse name in the mapping properties.
Pre SQL	The pre-SQL command to run on the Snowflake source table before the agent reads the data. For example, if you want to update records in the database before you read the records from the table, specify a pre-SQL statement. The query must include a fully qualified table name. You can specify multiple pre-SQL commands, each separated with a semicolon.
Post SQL	The post-SQL command to run on the Snowflake table after the agent completes the read operation. For example, if you want to delete some records after the latest records are loaded, specify a post-SQL statement. The query must include a fully qualified table name. You can specify multiple post-SQL commands, each separated with a semicolon.
Table Name	Overrides the table name of the imported Snowflake source table.
SQL Override	The SQL statement to override the default query used to read data from the Snowflake source.
Tracing Level	Determines the amount of detail that appears in the log file. You can select Terse, Normal, Verbose Initialization, or Verbose Data. Default value is Normal.

Source objects and operations

In a Source transformation, you can use a single object, multiple objects, query, or parameter as the source type to read data from Snowflake.

Some restrictions apply with certain source objects.

Parameter source type

When you parameterize the source object and connection and enable the **Allow parameter to be overridden at run time** option in a transformation, you cannot override the object name using the fully qualified name such as `db.schema.tablename`.

You can pass the `db=<dbname>&schema<schemaname>` values in the **Additional JDBC URL Parameters** field in the Snowflake Data Cloud connection.

Multiple source type

You can use a single Source transformation to read from multiple Snowflake tables within the same database. To read from multiple Snowflake sources, you can create multiple Source transformations and then use a Joiner transformation to join the sources.

To read from multiple tables using a single Source transformation, select multiple object as the source type and then configure a join to combine the tables. You can either add related objects with PK-FK relationships that are already defined or you can define a relationship condition to join the tables. To set your own conditions to define the relationship between the tables, select **Advanced Relationship** from the **Related Objects Actions** menu, and then define the relationship. When you configure a join expression, select the fields and define a join query syntax. You must specify only the condition and not the type of join in the query. The condition you specify in the text box for the expression is appended to the join condition.

When you specify a join condition in the advanced relationship to join the tables, you cannot override the database and schema names from the connection. You need to manually change the database and schema name in the advanced relationship condition. If the condition includes columns with a fully qualified name such as `db.schema.tablename`, do not configure an override. Delete the fully qualified database and schema names from the advanced relationship condition and then run the mapping.

Restrictions for the multiple source type

A multiple source type has the following restrictions:

- A mapping fails when you read data from multiple tables joined using related objects and the tables and column names have case-sensitive columns.
- A mapping configured with a join for one or more tables that have the same column names fails.
- A mapping that reads from multiple Snowflake objects that do not belong to the same database and schema fails.
- A mapping configured with a join that reads from multiple tables fails if you specify an override to the table, schema, or database in the Snowflake source advanced properties.

Query source type

When you use a custom SQL query to import Snowflake tables, specify the Snowflake database and schema name in the custom SQL query. If you do not specify the database and schema name, the agent considers the database and schema name specified in the connection properties. The table name in the query that reads from Snowflake must be a fully qualified. When you use a custom query to call a stored procedure, ensure that the role has access to the database and schema.

You can use a query source type with the following restrictions:

- Do not configure an override for the database and schema names in the connection.

- The following operations are not applicable for the query source type:

  - Filter and sort options.

  - Source partitioning.

  - Advanced properties, except for pre-SQL and post-SQL statements.

  - In advanced mode, you cannot call a stored procedure from a custom query in a Source transformation.

- Do not configure nested in-out parameters in the query source type.

- When you use a parameterized query source type in a mapping to read from multiple tables, and you configure a join for one or more tables that have the same column names, the mapping fails.
  For example, see the following SQL query that involves a full outer join between two tables EMPLOYEE and DEPARTMENT that are part of the SALES.PUBLIC schema, where two columns have the same name, CITY:

    ```
    SELECT EMP_ID, NAME, CITY, DEPT_ID, DEPT_NAME, CITY FROM SALES.PUBLIC.EMPLOYEE
    FULL OUTER JOIN SALES.PUBLIC.DEPARTMENT ON EMP_ID = DEPT_ID
    ```

  To distinguish the conflicting column names, add aliases that the database can refer to while joining the tables:

    ```
    SELECT e.EMP_ID, e.NAME, e.CITY as ecity,d.DEPT_ID, d.DEPT_NAME, d.CITY as dcity FROM
    SALES.PUBLIC.EMPLOYEE e
    FULL OUTER JOIN SALES.PUBLIC.DEPARTMENT d ON e.EMP_ID = d.DEPT_ID
    ```

General restrictions for the source

The Source transformation has the following general restrictions:

- You cannot read or write Snowflake table and column names that contain double quotes. The mapping fails with the following error: `SQL compilation error`

- In advanced mode, mappings fail to read from more than 500 columns and result in the following error: `HTTP POST request failed due to IO error.`

- You cannot use system variables in filters.

- When you configure multiple filter conditions and one of the conditions includes an IS NULL operator, ensure that you place the condition with the IS NULL operator at the end. Otherwise, the mapping fails.

- When you import a Snowflake object, ensure that the schema names do not contain a slash (/) or backslash (\). Otherwise, tables in the schema with a name that contains a slash (/) or backslash (\) do not reflect in the user interface in the mapping object.

Key range partitioning

You can configure key range partitioning when you use a mapping to read data from Snowflake sources. The partition type controls how the agent distributes data among partitions at partition points. Partitioning is not applicable for mappings in advanced mode.

Partitioning optimizes the mapping performance at run time. When you run a mapping configured with key range partitioning, the agent distributes rows of source data based on the field that you define as partition keys. The agent compares the field value to the range values for each partition and sends rows to the appropriate partitions.

Click **Add New Key Range** to define the number of partitions and the key ranges based on which the agent must partition data.

The following image shows an example of the configured partitioned ranges on the **Partitions** tab:
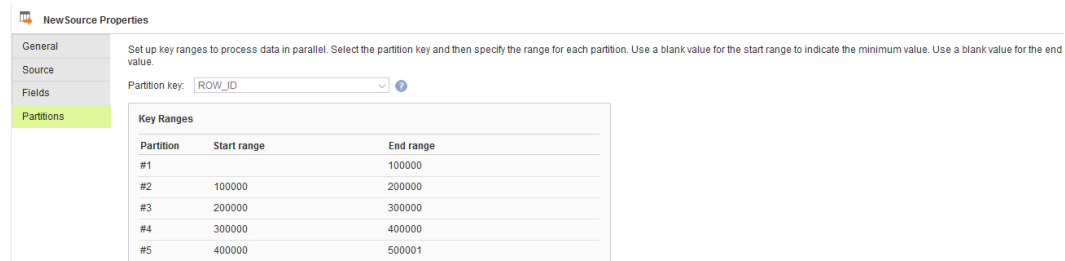
Use a blank value for the start range to indicate the minimum value. Use a blank value for the end range to indicate the maximum value.

Use key range partitioning for columns that have an even distribution of data values. Otherwise, the partitions might have unequal size. For example, a column might have 10 rows between key values 1 and 1000 and the column might have 999 rows between key values 1001 and 2000. If the mapping includes multiple sources, use the same number of key ranges for each source.

When you define key range partitioning for a column, the agent reads the rows that are within the specified partition range. For example, if you configure two partitions for a column with the ranges as 10 through 20 and 30 through 40, the agent does not read the rows 20 through 30 because these rows are not within the specified partition range.

You can configure a partition key for fields of the following data types:

- Integer

- String

- Any type of number data type. However, you cannot use decimals in key range values.

- Datetime. Use the following format to specify the date and time: YYYY-MM-DD HH24:MI:SS. For example, 1971-01-01 12:30:3
  If you specify the date and time in any other format, the task fails.

When you run a mapping enabled for key range partitioning, you can't filter source data even though you configure a filter in the Source transformation.

Overriding SQL

When you override the object, database, schema, or role, you must follow some guidelines.

If you specify an SQL override query to override the custom query used for importing the metadata from Snowflake tables, specify a fully qualified table name.

When you specify both an SQL override and overrides to the database, schema, warehouse, or role from the advanced source properties, consider the following guidelines for the override combinations:

Override the SQL and role

If you override the SQL and the role of a Snowflake object, verify that the override role has access to the table selected in the mapping.

To override the SQL and the role, you can perform one of the following tasks:

- Grant the overriding role with the same access permissions as the role used for the Snowflake object that you selected in the mapping.
- Specify an override to the table from the advanced properties. The specified override table is used in the design time and the SQL override takes precedence at runtime.

Override the SQL, database, schema, and role

If you select a Snowflake object in a mapping and you specify both an SQL override and also override the database, schema, and role from the advanced properties, and the table does not exist in the overriding database, the mapping fails with a `table1 not found` error.

Specify a valid table name that corresponds to the overriding database and schema name.

SQL override ports

The SQL override ports must be the same as the connected ports in the field mapping.

CHAPTER 5

Targets for Snowflake Data Cloud

Add a Target transformation to write data to a target. When you add a Target transformation to a mapping, you define the target connection, target objects, and target properties related to the Snowflake Data Cloud connection type.

Target properties for Snowflake Data Cloud

You can configure a Target transformation to represent a Snowflake Data Cloud target.

The following table describes the Snowflake Data Cloud target properties that you can configure in a Target transformation:

Property	Description
Connection	Name of the target connection. You can select an existing connection, create a new connection, or define parameter values for the target connection property. **Note:** You can switch between a non-parameterized and a parameterized Snowflake Data Cloud connection. When you switch between the connections, the advanced property values are retained. To overwrite the target connection properties at runtime, select the **Allow parameter to be overridden at run time** option.
Target Type	Type of target object. Select **Single Object** or **Parameter**.
Parameter	A parameter file where you define values that you want to update without having to edit the task. Select an existing parameter for the target object or click **New Parameter** to define a new parameter for the target object. The **Parameter** property appears only if you select parameter as the target type. If you want to overwrite the target object at runtime, select the **Allow parameter to be overridden at run time** option. When the task runs, it uses the parameters from the file that you specify in the task advanced session properties.
Object	The target object for the task. Select the target object. You can either select an existing table or create a new table. You can write data by selecting an existing table or creating a new table in the target by using the **Create New at Runtime** option.

Property	Description
Create New at Runtime	Creates a Snowflake target table at runtime based on the table type and the path you specify. To create a target table at runtime, provide the following parameters: - Optional. Specify the table type as `table`. - In the **Path** field, specify the Snowflake database name and schema in the following format: `<database name>/<schema>` The agent creates the target table based on the object name and the path you specify. **Note:** You can edit the metadata of the source fields before creating the target.
Use Exact Source Field Names in Target	Applies to the **Create New at Runtime** option. Determines if you can create a target object at runtime with the exact source field names. Select to retain all the source field names in the target exactly as in the source, including any special characters. If you disable this option, special characters in the source are replaced with underscore characters in the target. Default is disabled.
Operation	The target operation. Select Insert, Update, Upsert, Delete, or Data Driven.
Update columns	The primary key column to update, upsert, or delete data in a Snowflake target. If you do not specify a primary key column, the mapping considers the target column that you configured as the primary key column to update, upsert, or delete data.
Data Driven Condition	Enables you to define expressions that flag rows for an insert, update, upsert, or delete operation.

The following table describes the advanced properties that you can configure in a Target transformation:

Advanced Property	Description
UpdateMode	Loads data to the target based on the mode you specify. This property applies when you select the **Update**, **Upsert**, or **Data Driven** operation with the update or upsert condition. Select one of the following modes: - Update As Update. Updates records in the target table if the specified primary key column value matches with the incoming column value. - Update Else Insert. Updates records in the target table if the specified primary key column value matches with the incoming column value. If the primary key column value does not match, the mapping inserts a new row with the records.
Database	Overrides the database that you used to import the object.
Schema	Overrides the schema that you used to import the object.
Warehouse	Overrides the Snowflake name specified in the connection. The warehouse name in the mapping overrides the warehouse name you specify in the connection. Even though you provide an incorrect warehouse name in the connection properties, the connection is successful. However, before you run the mapping, ensure that you specify the correct warehouse name in the mapping properties.
Role	Overrides the Snowflake role assigned to the user specified in the connection.

Advanced Property	Description
Pre SQL	The pre-SQL command to run before the agent writes to Snowflake. For example, if you want to assign sequence object to a primary key field of the target table before you write data to the table, specify a pre-SQL statement. You can specify multiple pre-SQL commands, each separated with a semicolon.
Post SQL	The post-SQL command to run after the agent completes the write operation. For example, if you want to alter the table created by using create target option and assign constraints to the table before you write data to the table, specify a post-SQL statement. You can specify multiple post-SQL commands, each separated with a semicolon.
Batch Row Size[1]	The number of rows written to a single file in the agent location. When the number of rows written to the file reaches the value specified, the agent flushes the data queue and starts processing the write commands. For more information on configuring the batch size value, see "Configuring the batch size and the number of local staging files" on page 51
Number of local staging files[1]	The number of files that represents a single batch of data. The default number of files considered is 64. After the agent uploads the specified number of local staging files to the Snowflake user stage, Snowflake unloads the data to the target table. For more information on configuring the number of local staging files, see "Configuring the batch size and the number of local staging files" on page 51.
Truncate Target Table	Truncates the database target table before inserting new rows. Select one of the following options: - True. Truncates the target table before inserting all rows. - False. Inserts new rows without truncating the target table Default is false.

Additional Write Runtime Parameters	The additional runtime parameters that you can use when you write to Snowflake.
	You can enter multiple write runtime parameters, separated by ampersand (&), in the following format:
	`<param1>=<value>&<param2>=<value>&<param3>=<value>....`
	For example, if you want to specify the user-defined stage in the Snowflake database to upload the local staging files and don't want to compress files before you write to Snowflake tables, enter the following runtime parameters:
	`remoteStage=REMOTE_STAGE&Compression=Off`
	For the list of additional write runtime parameters that you can configure, see "Write runtime parameters" on page 47.

Table Name	Overrides the table name of the Snowflake target table.
Rejected File Path[1]	The filename and path of the file on the agent machine where you want to write the rejected records. For example, `\rejectedfiles\reject7`
Update Override	Overrides the default update query that the agent generates for the update operation with the update query.
Success File Directory	Not applicable.

Advanced Property	Description
Error File Directory	Not applicable.
Forward Rejected Rows	Determines whether the transformation passes rejected rows to the next transformation or drops rejected rows. By default, the agent forwards rejected rows to the next transformation.
[1]Doesn't apply to mappings in advanced mode.	

Write runtime parameters

You can customize and set any additional runtime parameters when you write to Snowflake.

If you specify more than one additional write runtime parameter, separate each key-value pair with an ampersand (&).

You can configure the following properties as additional write runtime parameters in the Target transformation:

- To optimize the write performance, you can choose to compress files before writing to Snowflake tables. You can set the compression parameter to **On** or **Off**. By default, compression is on.
  For example, if don't want to compress files before you write to Snowflake tables, enter the additional parameter in the following format:

  ```
  Compression=Off
  ```

- To replace values specified in fields and write them as Null values to a target, enter the additional parameter in the following format:
  ```
  null_if=('Field value1','Field value2','Field value3','...')
  ```

  When you specify more than one field value, separate each field value with a comma (,).

  This parameter doesn't apply to mappings in advanced mode.

- To trim the leading and trailing white spaces from strings when you write to a target, enter the additional parameter in the following format:
  ```
  trim_space=true
  ```

  This parameter doesn't apply to mappings in advanced mode.

  You can't trim spaces from strings in the following scenarios:

  - When strings are enclosed with double quotes.

  - When you enable the staging property for the write operation.

For more information about these additional parameters, see <u>Copy into table</u> in the Snowflake documentation.

Target objects and operations

In a Target transformation, you can use a single object or parameter as the target type.

When you choose the target type, you can select the operation to insert, update, upsert, or delete data in a Snowflake target. You can also use the data driven operation to define expressions that flag rows for an insert, update, delete, or reject operation.

Target operation restrictions

Some rules apply when you configure a target operation.

Parameterized target

When you parameterize the target object and connection and enable the **Allow parameter to be overridden at run time** option in a transformation, you cannot override the object name using the fully qualified name such as `db.schema.tablename`. You must pass the `db=<dbname>&schema<schemaname>` values in the **Additional JDBC URL Parameters** field in the Snowflake Data Cloud connection.

Update columns

You need to specify the temporary key column to update data to or delete data from a Snowflake target. If the Snowflake target does not include a primary key column, click **Add** to add a temporary key. You can select multiple columns.

If the records from the source tables contain duplicate primary keys, perform one of the following tasks in mappings to update or delete records in Snowflake:

- Before you import the target table, define multiple primary keys in the target table.
- Define more than one custom key for the target object using the Update Columns option in the advanced target properties.

When you configure a mapping in advanced mode, you must additionally consider the following guidelines:

- When you specify an expression for a data driven condition and do not specify a column in the **Update Columns** field, a validation message does not appear. For DD_INSERT and DD_REJECT, the update column is not mandatory. For other operations such as DD_UPDATE and DD_DELETE, select at least one field in the update columns for the operation to work.
- When you use a parameterized Snowflake connection in a Target transformation, the **Update Columns** field does not display in the target properties. In the Target transformation, select a valid connection and object that display in the list and then select the operation.
- When the Snowflake target object in the mapping is parameterized and the selected operation is data driven or upsert, the Update Column field does not display in the dynamic mapping task target properties.

Data driven operation

When you configure a data driven expression to update the rows in Snowflake from a mapping in advanced mode, you need to specify the third argument in the IIF condition. If you do not specify a third argument in the IIF condition, the agent treats the operation as an insert for all non-matching rows.

For example, if you give a condition `IIF(Update_column=2,DD_UPDATE)` without including the third argument, such as `DD_DELETE` in the following expression `IIF(Update_column=2,DD_UPDATE,DD_DELETE)`, for any other row where the value of the update_column is not equal to 2, the agent performs an insert by default.

In the example, from the expression `IIF(COL_INTEGER = 2, DD_UPDATE)`, COL_INTEGER=2 resolves to 1 and COL_INTEGER!=2 resolves to 0, where 1 is the internal ID for the Update operation and 0 is the internal ID for the Insert operation. The value of the third argument when not specified defaults to 0.

You can use data driven operation with the following restrictions for mappings in advanced mode:

- If one of the source rows that is marked for Insert is already available in the target, the agent still inserts the row to the target.
- When you define an expression for a Data Driven operation type, the expression editor displays column names up to 120 characters only.
- If the expression in the data driven condition contains special characters, the agent fails to validate the data driven condition and the mapping runs successfully.

General restrictions

In advanced mode, mappings that write data to more than 500 columns fail with the following error: `HTTP POST request failed due to IO error`

Passthrough partitioning

You can configure partitioning to optimize the mapping or performance at run time when you write data to Snowflake targets.

The partition type controls how the agent distributes data among partitions at partition points. You can define the partition type as passthrough partitioning. With partitioning, the agent distributes rows of target data based on the number of threads that you define as partition.
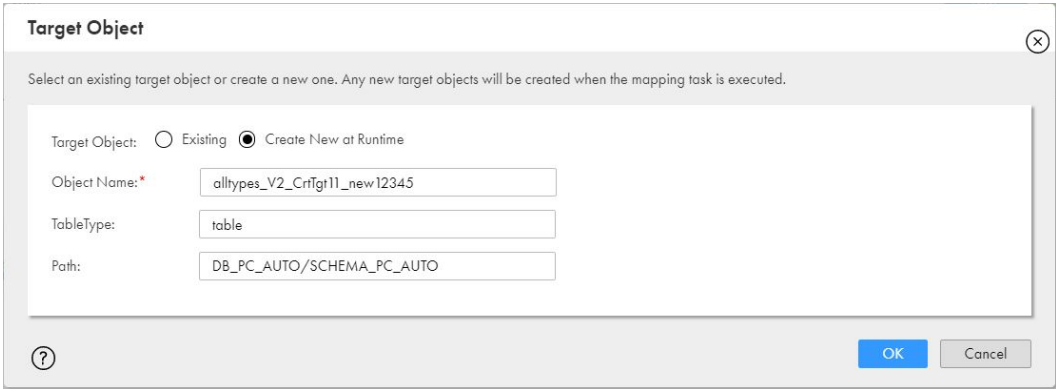
Specify a target

You can use an existing target or create a new target to write data to Snowflake. If you choose to create a new target, the agent creates the target when it runs the task.

Ensure that the path for the schema and database that you specify for the target object is in uppercase letters. Specify the Snowflake database name and schema in the following format: <database name>/ <schema name> for the target object is in uppercase letters. If you do not enter the path, the Secure Agent considers the schema and database name you specified in the **Additional JDBC URL Parameters** field from the Snowflake Data Cloud connection properties.

Specify the TableType as `table`. The TableType property is optional.

The following image shows an example of a target object configured to create at runtime:

Restrictions for a Snowflake target

The following rules apply when you configure the target:

- If the Secure Agent is installed on Windows, you cannot write data to a Snowflake target table when the table names contain the following special characters: /\:*?"<>|

- You can create a target at runtime with the following restrictions:
  - The table and column names are case sensitive. The Secure Agent adds double quotes to the table and column name while creating the target table at runtime. It is mandatory that you query the table name and column name in Snowflake using double quotes.
  - When you add an advanced filter condition for the source object, or when you define a relationship for multiple source objects in a Source transformation, if the condition contains the "/" separator to separate the database name, schema name, and table name, the mapping fails with an SQL compilation error. You must change the slash "/" separator to a dot (.) in the SQL query and run the mapping.
  - If the target name you specify contains special characters and you do not enable the target to use the exact source field names in the target, and the mapping also contains an Expression or Aggregator transformation, the target table is created with the special characters. The rest of fields that contain special characters that comes from the upstream transformations are replaced with underscore characters in the target.
  - When you parameterize the target object in a mapping and you create a target at runtime, the target object must have the full path in the parameter file. However, if you use an existing target object, the target object must have only the table name.

Override the update operation

You can specify an update override to override the update query that the Secure Agent generates for the update operation.

When you configure an update override, the Secure Agent uses the query that you specify, stages the data in files, and then loads that data into a temporary table using the Snowflake's loader copy command. The data from the temporary table is then loaded to the Snowflake target table. The syntax that you specify for the update query must be supported by Snowflake.

Specify the update query in the following format:

```
UPDATE <Target table name> SET <Target table name>.<Column1> = :TU.<Column1>, <Target
table name>.<Column2> = :TU.<Column2>, … <Target table name>.<ColumnN> = :TU.<ColumnN>
FROM :TU WHERE <Target table name>.<Update Column1> = :TU.<Update Column1> AND <Target
table name>.<Update Column2> = :TU.<Update Column2> AND … <Target table name>.<Update
ColumnN> = :TU.<Update ColumnN>
```

where, *:TU.* represents the incoming data source for the target port.

The Secure Agent replaces *:TU.* with a temporary table name while running the mapping and does not validate the update query.

When you configure an update override in a mapping to write to Snowflake, consider the following rules:

- Ensure that the column names for :TU matches the target table column names.
- Ensure that the column names are fully qualified names.
- Specify the update query with a valid SQL syntax because Snowflake Data Cloud Connector replaces :TU with a temporary table name and does not validate the update query.
- Do not change the order of the column in the mappings when you configure the update override option.
- The update query in the mapping must not contain unconnected fields to the target.
- Ensure that the **UpdateMode** advanced property is set to **Update As Update**.

Optimize the .csv file size

When you create a mapping to write to Snowflake, you can specify the size of the local staging .csv file in bytes. Specify the local staging file size property, csvFileSize, in the **Additional Write Runtime Parameters** field in the advanced Snowflake target properties. The default file size is 50 MB.

If the intended file size is 50 MB, calculate the csvFileSize value in bytes, for example 50*1024*1024 and then specify 52428800 as the csvFileSize. For optimal performance, the number and size of the CSV files must be in synchrony.

Configuring the batch size and the number of local staging files

To achieve optimal performance of a mapping that writes data to Snowflake, allow the mapping to consider the default values for both the batch size and number of local staging files based on the amount of data it processes. Unless your use case requires that you set the values, it is recommended that the mapping processes with the default values.

The priority that the mapping gives to these two configurations depends on which threshold is reached first. If the mapping satisfies the criteria for the number of local staging files before reaching the specified batch row size, the number of local staging files takes precedence since that threshold is reached first.

For example, see the following table which displays the number of insert commands issued for various configurations of the specified batch size and the number of local staging files in a write operation to Snowflake:

Number of Source Rows	Row Size (bytes)	Default File Size	Batch Row Size	Number of Local Staging Files	Number of Inserts
4	52019206 bytes	52428800 bytes	4	1	2
4	52019206 bytes	52428800 bytes	4	Not configured. Considers the default value of 64.	1
4	52019206 bytes	52428800 bytes	1	Not configured. Considers the default value of 64.	4
4	52019206 bytes	52428800 bytes	3	1	2
4	52019206 bytes	52428800 bytes	1	1	4

Scenario 1: The batch row size is 4 and the number of local staging files is 1.

With a batch row size of 4 and 1 local staging file, two insert statements are executed in Snowflake when processing data.

When the file size exceeds the default 50 MB because there are two rows, a new file is created for the next incoming row. Given that the number of local staging files is set to 1, a stage can hold a maximum of one file. Hence, two stages are created, each holding one file with two rows. Each of the file sizes is 104038412 bytes, which is 52019206 bytes per row. For each stage, an insert command is issued, leading to two insert commands.

Scenario 2: The batch row size is 4 and the number of local staging files value is not set.

Using the default value of 64 for the local staging files, the mapping considers the batch size when the local staging file threshold is not reached.

Scenario 3: The batch row size is 1 and the number of local staging files value is not set.

With a batch row size of 1 and no specific local staging files value set, the mapping issues four insert statements for four rows as the staging files threshold is not reached.

Scenario 4: The batch row size is 3 and the number of local staging files value is 1.

Using a batch row size of 3 and one local staging file, the mapping issues two insert commands after processing two rows in the first stage and two rows in the second stage.

Scenario 5: The batch row size is 1 and the number of local staging files value is 1.

With a batch row size of 1 and one local staging file, four insert commands are issued for each row individually as the staging files threshold is never met due to the specified smaller row size.

Configure load properties in mappings

You can configure write properties to load data to Snowflake in the **Additional Write Runtime Parameters** field in the Snowflake Data Cloud advanced target properties of the Target transformation.

The following table lists some of the additional runtime parameters that you can specify to load data to Snowflake:

Property	Description
oneBatch	Process all data in a single batch. Type is Boolean. Default is false.
remoteStage	Specifies to use internal stage. External stage is not applicable. Type is String. Default is "~"(user stage).
onError	Specifies the action to perform when an error is encountered while loading data from a file. For example, `onError option ABORT_STATEMENT|CONTINUE|SKIP_FILE` Type is String. Default is CONTINUE.
compressFileByPut	Compress file by PUT. Type is Boolean. Default is false.
compressDataBeforePut	Compress data before PUT. The loader compresses the data to a gzip format before uploading the data. Type is Boolean. Default is true.
copyEmptyFieldAsEmpty	The COPY command option to set incoming empty fields as null. Type is Boolean.

Property	Description
enablePGZIP	Enables parallelism for the file compression. Type is Boolean. Default is true.
onError=ABORT_STATEMENT&oneBatch=true	Load the entire data in single batch and to stop the task if an error occurs. Simultaneously, validate the user-specified reject file path and write the error records to this file and to the session log. Type is onError - String or oneBatch - Boolean.

When you set the values in the additional runtime parameters field, every configured partition initializes a new loader instance and the configured values apply similarly across all the partitions.

Example 1. Compress files

You want to compress files by using the Put command before loading data to Snowflake.

Specify the following compression option: `compressDataBeforePut=false&compressFileByPut=true`

If you specify both the options as true, Snowflake considers the `compressDataBeforePut` option.

Example 2. Replace empty values as null

You want to replace the incoming fields with empty values as NULL while loading the data to Snowflake.

Specify the `copyEmptyFieldAsEmpty Boolean` option and set the value to true or false based on your requirement.

Consider the following scenarios before you configure the `copyEmptyFieldAsEmpty Boolean` parameter:

- If you do not configure this parameter, Null values are received as NULL, and empty values are received as Empty. This is the default behavior.

- If you set the parameter `copyEmptyFieldAsEmpty=false`, Null values as received as Null and empty values are received as Null.

- If you set the parameter `copyEmptyFieldAsEmpty=true`, Null values are received as empty, while empty values are received as empty.

Example 3. Write data in a single batch

You want to write source data in a single batch to Snowflake. If an error is encountered while loading, you also want to capture the errors.

Specify the `onError=ABORT_STATEMENT&oneBatch=true` property based on your requirement.

While loading in a single batch, if an error occurs, the Secure Agent checks for the specified reject file name, runs the COPY command, validates the reject file, and then passes the file name to capture the errors, if any.

Configure additional runtime parameters to run mappings in advanced mode

In advanced mode, you can configure additional properties to write data to Snowflake. Specify the parameters in the **Additional Write Runtime Parameters** field in the Target transformation.

You can configure the following additional runtime parameters to write data to Snowflake:

autopushdown

Optional. Determines whether the automatic query SQL ELT is enabled.

If you enable SQL ELT and the query runs on an advanced cluster, the cluster application pushes part of the query to process in Snowflake, thereby optimizing the performance of these queries.

Default is *on* when the connector uses a compatible Spark version. When the connector does not use a compatible Spark version, the default value is *off*.

continueOnError

Optional. Determines whether the COPY command aborts the operation when you enter data that is not valid. For example, you specify a JSON format for a variant data type column that is not valid.

The values include *on* and *off*. When you specify the value as *on*, the COPY command continues even if an error occurs. If you specify *off*, the COPY command aborts when an error occurs. Default is *off*.

It is recommended that you keep the option as *off*. Else, when an error is encountered while copying data into Snowflake, some of the data might be missing.

parallelism

The size of the thread pool to use when the Secure Agent uploads or downloads data between Snowflake and the advanced cluster. Default is 4.

Do not change the default value unless you need to increase or decrease the throughput. When you want a high throughput, do not set the parallelism to an arbitrarily large number. A high value of parallelism might lead to undesired outputs and slows down the operation.

purge

Determines whether the Secure Agent deletes the temporary files created when transferring data from an advanced cluster to Snowflake through the external data transfer. The possible values are *on* and *off*. Default is *off*.

If you set this parameter to *off*, the Secure Agent automatically deletes the temporary files. Purging works only for data transfers from an advanced cluster to Snowflake, but not for transfers from Snowflake to the advanced cluster. If you set this parameter to *on*, the Secure Agent does not automatically delete the temporary files.

usestagingtable

Optional. Determines whether the data loading operation uses a staging table.

Snowflake creates a staging table with a temporary name. If the data loading operation is successful, Snowflake drops the original target table and renames the staging table to the original target table name. If the data loading operation fails, Snowflake drops the staging table and the target table retains the data that it contained before the operation.

Snowflake strongly recommends that you use a staging table. To create a staging table, you must have sufficient privileges to run the COPY command to create a table. If you do not have permissions to create a table, you can load directly without using a staging table.

The values include *on* and *off*. If you specify the **usestagingtable** parameter as *on*, Snowflake uses a staging table. If you specify the value as *off*, Snowflake directly loads the data into the target table. Default is *on*.

Capturing changed data from CDC sources

You can use Snowflake Data Cloud Connector to write changed data from a CDC source such as Oracle CDC and Oracle CDC V2 and write the changed data to a Snowflake target.

When you configure a mapping, add the CDC sources and then run the associated mapping task to write the changed data to Snowflake. If you define a column as required in the Snowflake target table, map a column in the CDC source to the required column in the Snowflake target in the mapping before you run the task.

When the mapping task processes the changed data from a CDC source, Snowflake Data Cloud Connector creates a state table in Snowflake. When the changed data is received from the CDC source, Snowflake Data Cloud Connector uploads the changed data to the staging table. Then, it generates a `Job_Id` and writes the `Job_Id` to the state table along with the restart information.

The connector then merges the stage table with the actual target table in Snowflake. Each time you run the mapping task, Snowflake Data Cloud Connector creates the state table, if it does not exist, to store the state information.

Snowflake Data Cloud Connector uses the following naming convention for the tables:

- State table name: `<MappingTaskID>_<Instance(s)>`
- Staging table name: `<MappingTaskID>_<TargetInstanceName>`

Restrictions

Consider the following restrictions when you capture changed data from CDC sources:

Mapping

You can use a Snowflake target in a CDC mapping to write data from CDC sources. You cannot use a Snowflake source or lookup in a CDC mapping.

staging optimization property

The optimization property that you set for staging data in the DTM of the Secure Agent is not applicable. If you run a mapping with both CDC and staging property enabled, the mapping runs successfully. However, staging optimization is disabled and you can view a message logged in the session logs.

Recovery initialization error

When you run a CDC mapping to write data from a CDC source to a Snowflake target created at runtime, you might encounter the following error:

```
Error occured while initializing CCI State Handler
com.informatica.cci.runtime.internal.utils.impl.CExceptionImpl: Internal error: Recovery
Init failed
```

To avoid this error, you must have the grant permissions to create a table in Snowflake.

DTM error

When you run a CDC mapping that uses a Snowflake target connection configured with the ProcessConnDB and ProcessConnSchema parameters in the **Additional JDBC URL Parameters** field, the mapping might fail with the following DTM error:

```
Internal error. The DTM process terminated unexpectedly. Contact Informatica Global
Customer Support.
```

To avoid this error, remove the ProcessConnDB and ProcessConnSchema parameters from the **Additional JDBC URL Parameters** field in the Snowflake Data Cloud connection. However, if you want to create temporary stage tables and recovery state tables using the ProcessConnDB and ProcessConnSchema parameters, verify that you have the necessary permissions in the required database and schema.

Configuring a mapping task to read from a CDC source

You can use Snowflake Data Cloud Connector to capture changed data from a CDC source and write the changed data to a Snowflake target.

Add the CDC source in the mapping, and then run the associated mapping task to write the changed data to the Snowflake target. You can also configure multiple pipelines in a single mapping to write the captured changed data to a Snowflake target.

When you configure a mapping to write changed data from a CDC source to Snowflake, you can configure the following advanced properties in the Snowflake Target transformation:

- Database
- Schema
- Warehouse
- Role
- Pre SQL
- Post SQL
- Truncate target table
- Table name
- Update override

Step 1. Configure the source

Configure a Source transformation to read from a CDC source such as Oracle CDC and Oracle CDC V2.

1. In the Source transformation, specify a name and description in the general properties.
2. In the **Source** tab, select any configured CDC connection and specify the required source properties.

   It is recommended that the source object contains a primary key.

Step 2. Configure the target

Configure a Target transformation to write changed data from a CDC source to Snowflake.

You can only configure a single Snowflake Data Cloud target transformation in a mapping to write changed data from a CDC source.

1. On the **Target** tab, perform the following steps to configure the target properties:
   a. In the **Connection** field, select theSnowflake Data Cloud connection.
   b. In the **Target Type** field, select the type of the target object.
   c. In the **Object** field, select the required target object.
   d. In the **Operation** field, select **Insert** or **Data Driven**.

      Note: Update, upsert, and delete target operations are not applicable. Ensure that the target tables do not have a primary key defined in Snowflake.
   e. If you select **Data Driven Condition**, specify the **DD_INSERT** condition.

      Note: Ensure that the target tables do not have a primary key defined in Snowflake.
   f. Configure the applicable advanced target properties for the CDC mode.

2. On the **Field Mapping** tab, map the incoming fields to the target fields. You can manually map an incoming field to a target field or automatically map fields based on the field names.

   If you define a column as required in the Snowflake target table, map a column in the CDC source to the required column in the Snowflake target in the mapping.

Step 3. Configure the mapping task

After you create a mapping, add the mapping to a mapping task, and configure the advanced properties. Run the associated mapping task to write the changed data to Snowflake.

1. From the **Actions** menu, click **New Mapping Task**.

   The **New Mapping Task** page appears.

2. In the **Definition** tab, enter the task name and select the configured mapping.

3. In the **CDC Runtime** tab, specify the required properties for the selected CDC source.

   For more information about the **CDC Runtime** properties, see the source properties for the selected CDC source.

4. In the **Runtime Options** tab, add the following properties in the **Advanced Session Properties** section:

   a. In the **Commit on End of File** field, select the value of the property as **No**.

   b. In the **Commit Type** field, select the value of the property as **Source**.

   c. In the **Recovery Strategy** field, select the value of the property as **Resume from last checkpoint**.

5. Click **Save** > **Run** the mapping task.

   Alternatively, you can create a schedule that runs the mapping task on a recurring basis without manual intervention. You can define the schedule to minimize the time between mapping task runs. In **Monitor**, you can monitor the status of the logs after you run the task.

To improve performance, specify a higher commit interval for the Maximum Rows Per Commit property in the CDC Runtime page in the mapping task wizard. However, in case of failure, recovery takes more time for a higher commit interval.

Disabling the recovery mechanism

If you do not want to retrieve changes using the recovery state table when a CDC job encounters a failure, you can disable recovery in the target and mapping task.

To disable recovery, perform the following tasks:

1. Set the *disableSnowflakeRecoveryCDC=true* property in the **Additional Write Runtime Parameters** field in the Snowflake Data Cloud Target transformation.

2. In the **Advanced Session Properties** on the **Runtime Options** tab in the mapping task, remove the configured recovery strategy.

Note: Ensure that the source is not Snowflake in a CDC mapping. If the mapping contains a Snowflake source and target, and if you set the *disableSnowflakeRecoveryCDC* property for the target, the mapping job shows incorrect results.

Viewing job statistics

You can view statistics for each job to see the number of rows processed and the job state.

The job can have one of the following states:

- Success. The Secure Agent applied all rows of insert, update, and delete operations.

- Warning. The Secure Agent rejected one or more rows. The **Rows Processed** field in the **My Jobs** page reflects the total number of rows that the Secure Agent processed.

- Failed. The job did not complete because it encountered errors.

The following image shows the **My Jobs** page that shows the details of the state and the number of processed rows of a Snowflake Data Cloud job:

To view how many among the processed rows were a success and how many resulted in an error, select the specific instance name and view the **Results** section. You can view the number of success rows and error rows.

The following image shows the details of the Snowflake Data Cloud task:

You can download the session log to get details of the number of output rows, affected rows, applied rows, and rejected rows.

You might also encounter the following scenarios of target statistics for Snowflake Data Cloud write operations:

Constraint violation

In insert, update, or delete operation scenarios where the Secure Agent rejects rows due to a constraint violation, a warning appears in the **Job Properties** page. You can download the session log to view the target statistics.

Can't find a match

In update or delete operation scenarios where the Secure Agent does not find a match for some records, that number does not reflect in the **My Jobs** page and the session log. For example, if there are 5 input rows and the Secure Agent updates only 4 target rows, the status of the number of processed rows stills reflects as 5. This issue occurs when Snowflake does not return an error message for rejected rows.

Non-unique match

In update or delete operation scenarios where the Secure Agent updates or deletes more rows because of a non-unique match, that actual number of updated or deleted records does not reflect both in the **My Jobs** page and in the session log. For example, if there were 5 input records and the Secure Agent updated 10 target rows, the **My Jobs** page reflects only 5 processed rows.

Success rows not updated

The number of success rows for the target object in the **Job Properties** page is not updated and remains the same as the number of incoming rows. For example, while writing 5 records to the target, if two records are rejected, the number of success rows still reflects as 5.

Rules and guidelines for Snowflake Data Cloud target transformations

The following rules and guidelines apply for Snowflake Data Cloud target transformations:

- If some records are rejected while writing large amounts of data to Snowflake, the rejected file might not display some of the rejected records even though the statistics of rejected records appear correctly in the session logs.
- If you run a mapping configured to create a new target at runtime, you can't write data to a Snowflake target when the table name contains a slash (/) or backslash (\).
- In advanced mode, the following restrictions apply:
  - The mapping fails when the Snowflake table names contains double quotes. You can include column names with double quotes.
  - When you specify a table name for a target that you want to create at runtime and you also specify an update override with a different table name, the mapping considers the table name specified during the design time.
  - You cannot call a stored procedure from a custom query from the Source transformation.
  - If the selected update column in a Target transformation contains the maximum length defined for the column name, the task fails. Ensure that the length of the column name length does not exceed 75 characters.

CHAPTER 6

Lookups for Snowflake Data Cloud

Add a Lookup transformation to retrieve data based on a specified lookup condition. When you add a Lookup transformation to a mapping, you define the lookup connection, lookup objects, and lookup properties related to Snowflake.

You can look up Snowflake data values based on a condition that you configure. In the Lookup transformation, select the lookup connection and object. Then, define the lookup condition and the outcome for multiple matches.

The mapping queries the lookup source based on the lookup fields and the defined lookup condition. The lookup operation returns the result to the Lookup transformation, which then passes the results downstream.

You can also set the default column value for the return field in a cached Lookup transformation in a Snowflake mapping. However, in advanced mode, you can't set the default column value for the return field.

You can configure the following lookups:

- **Connected.** You can use a cached or uncached connected lookup for mappings. You can also use a dynamic lookup cache to keep the lookup cache synchronized with the target.
- **Unconnected.** You can use a cached lookup. You need to supply input values for an unconnected Lookup transformation from a :LKP expression in a transformation that uses an Expression transformation.

Note: In advanced mode, cached and uncached lookups are not applicable.

For more information about Lookup transformation, see *Transformations* in the Data Integration documentation.

Lookup properties for Snowflake Data Cloud

The following table describes the Snowflake Data Cloud lookup object properties that you can configure in a Lookup transformation:

Property	Description
Connection	Name of the lookup connection. You can select an existing connection, create a new connection, or define parameter values for the lookup connection property. **Note:** You can switch between a non-parameterized and a parameterized Snowflake Data Cloud connection. The advanced property values are retained during the switch. If you want to overwrite the lookup connection properties at runtime, select the **Allow parameter to be overridden at run time** option.
Source Type	Type of the source object. Select Single Object, Query[1], or Parameter.
Parameter	A parameter file where you define values that you want to update without having to edit the task. Select an existing parameter for the lookup object or click **New Parameter** to define a new parameter for the lookup object. The **Parameter** property appears only if you select parameter as the lookup type. If you want to overwrite the parameter at runtime, select the **Allow parameter to be overridden at run time** option. When the task runs, it uses the parameters from the file that you specify in the task advanced session properties.
Lookup Object	Name of the lookup object for the mapping.
Multiple Matches	Behavior when the lookup condition returns multiple matches. You can return all rows, any row, the first row, the last row, or an error. You can select from the following options in the lookup object properties to determine the behavior: - Return first row [1] - Return last row [1] - Return any row - Return all rows - Report error
Filter	Not applicable.
Sort	Not applicable.

[1]Doesn't apply to mappings in advanced mode.

The following table describes the Snowflake Data Cloud lookup object advanced properties that you can configure in a Lookup transformation:

Advanced Property	Description
Database	Overrides the database specified in the connection.
Schema	Overrides the schema specified in the connection.

Advanced Property	Description
Warehouse	Overrides the Snowflake warehouse name specified in the connection. The warehouse name in the mapping overrides the warehouse name you specify in the connection. Even though you provide an incorrect warehouse name in the connection properties, the connection is successful. However, before you run the mapping, ensure that you specify the correct warehouse name in the mapping properties.
Role	Overrides the Snowflake role assigned to the user specified in the connection.
Pre SQL	Not applicable.
Post SQL	Not applicable.
SQL Override[1]	Overrides the default SQL query used to read data from the Snowflake source.
Lookup Data Filter[1]	Limits the number of lookups that the mapping performs on the cache of the lookup source table based on the value you specify in the filter condition. This property is applicable when you select object as the source type and enable lookup cache on the **Advanced** tab for the Lookup transformation. Maximum length is 32768 characters. For more information about this property, see *Transformations* in the Data Integration documentation.

[1]Doesn't apply to mappings in advanced mode.

Parameterization

When you parameterize lookup objects and connections, you must follow some rules to override the object name.

If you enable the **Allow parameter to be overridden at run time** option in a transformation, you cannot override the object name using the fully qualified name such as `db.schema.tablename`.

You must pass the `db=<dbname>&schema<schemaname>` values in the **Additional JDBC URL Parameters** field in the Snowflake Data Cloud connection.

Multiple match restrictions

When you configure a lookup, you define the behavior when a lookup condition returns more than one match. You can return all rows, any row, the first row, the last row, or an error.

The following configurations have multiple match policy restrictions:

- If the multiplicity option is set to Report error, the task does not fail.

- When you configure a lookup override without caching, the return first row and return last row multiple matches options are not applicable.

- Mappings configured for an uncached lookup and multiple matches fails when the Snowflake source contains case-sensitive tables and column names.

Enable lookup caching

When you configure a Lookup transformation in a mapping, you can cache the lookup data during the runtime session.

When you select **Lookup Caching Enabled**, Data Integration queries the lookup source once and caches the values for use during the session, which can improve performance. You can specify the directory to store the cached lookup. You can configure dynamic and persistent lookup caches.

For information about lookup caching, see the chapter "Lookup transformations" in *Transformations* in the Data Integration documentation.

Dynamic lookup cache has the following restrictions:

- When you configure dynamic lookup cache, set the **On Multiple Matches** property to **Report Error**. To reset the property, change the dynamic lookup to a static lookup, change the property, and then change the static lookup to a dynamic lookup. Ensure that all the lookup fields are mapped.

- A lookup condition in dynamic lookups can use only an equal operator.

- SQL ELT optimization is not applicable.

- You cannot parameterize a Lookup transformation enabled to use dynamic lookup cache.

Using ORDER BY and WHERE clause

You can use the ORDER BY clause in a query in a connected and unconnected cached lookup if you configure the `-DENABLE_SORTED_INPUT_FOR_LKP=true` property in the JVM options of the Secure Agent and enable the **Sorted Input** option in the advanced lookup properties.

Consider the following guidelines when you use the ORDER BY clause:

- When you configure a lookup condition to return all rows or any row, include the lookup fields in the ORDER BY statement.

- When you configure a lookup condition to return first or last row, additionally include the fields mapped in the target in the ORDER BY statement.

When you use an uncached connected lookup to read from and write to Snowflake and the data has null values, configure the `-DENABLE_NULL_FLAG_FOR_UNCACHED_LOOKUP=true` property in the JVM options of the Secure Agent to include the WHERE clause in the SQL statement.

Log uncached lookup queries

When you enable caching, Data Integration queries the lookup source once and caches the values for use during the session. When you disable caching, each time a row passes into the transformation, a SELECT statement gets the lookup values.

Data Integration logs the uncached lookup queries for a connected and unconnected lookup in the session log. Enable the **Lookup Caching Enabled** property in the mapping, and then enable the verbose mode in the Snowflake Data Cloud mapping task.

The following image shows the selected verbose mode in the mapping task:

When you run the mapping, Data Integration logs the uncached lookup queries in the session logs.

CHAPTER 7

Migrating a mapping

You can configure a connection and mapping in one environment and then migrate and run the mapping in another environment.

You can also migrate mappings configured in advanced mode. After the migration, you can change the connection properties from the Administrator service, but you do not need to modify the mapping. Data Integration uses the configured runtime attributes from the earlier environment to run the mapping successfully in the new environment.

Plan the migration

When you run the migrated mapping in the new environment, you might want to use the same or a different connection endpoint or object path. Based on your requirement, follow the guidelines in this section before you migrate a mapping.

The following table lists the configured advanced properties for an object type that you can retain when you migrate a mapping:

Object Type	Advanced properties
Single object	Database, schema, role, table
Multiple objects	Database, schema, role
Query	SQL override, role **Note:** The SQL override and custom query must be fully qualified.

You can also migrate mappings configured for dynamic schema handling.

Migrate a mapping within the same path

If you want the migrated mapping to use the same object path as in the earlier environment, you must maintain the same database, schema, and table in the Snowflake account for both the earlier and the new environment.

For example, if you have two different accounts, Account1 used for Organization 1 and Account2 used for Organization 2, the object path for the database, schema, and table name must be the same in both the accounts:

Account1: DB1/SCHEMA1/TABLE1

Account2: DB1/SCHEMA1/TABLE1

You do not need to override the database, schema, role, and table in the advanced properties. If the role used in the connection has access to TABLE1, the mapping runs successfully.

Migrate a mapping to a different path

You can use a different object path to run the mapping from the new environment.

Before you migrate the mapping, you can change the object metadata, runtime attributes, or the connection attributes to reflect the object path in the migrated environment. You do not have to edit or update the mapping in the new environment.

As a rule, when you specify the database, schema, or table in the advanced properties, connection, or object properties, Data Integration honors the attributes in the following order of precedence:

1. **Runtime advanced attributes**. The advanced properties such as database, schema, table, and role in the Source, Target, or Lookup transformation in a mapping.

2. **Connection attributes**. Attributes such as database and schema set in the **Additional JDBC URL Parameters** in the connection properties.

3. **Object metadata**. The object selected in the Source, Target, or Lookup transformation in a mapping.

Migration options

When you migrate to a different path, you can choose from one of the following options to update the object path:

Option 1. Update the connection properties to reference the new object

When you import the mapping from, for example, Org 1 into a new organization, for example, Org 2, you can change the existing connection in the **Review Connections** section to map to the connection that has access to the specified database, schema, and table in Org 2.

Option 2. Override the properties from the advanced properties

Before the migration, specify the required database, schema, table name, and role for the object in the new organization in the advanced properties of the Org 1 mapping.

After the migration, when you run the mapping, the Secure Agent uses the configured advanced parameters to override the object specified in the mapping imported from Org 1.

Option 3. Parameterize the properties in the mapping

You can choose to parameterize the advanced attributes, such as the database, schema, table name, and role before the migration. You can configure input parameters, in-out parameters, and parameter files in the mapping. When you use a parameter file, you can save the parameter file on a local machine or in a cloud-hosted directory. After you migrate the mapping, do not edit or update the mapping. If you have used in-out parameters for the advanced attributes such as for the database, schema, table name, and role, you can update these from the parameter file.

Parameterizing only the advanced properties, but not the object

If you want to parameterize only the advanced properties and use them at runtime, select a placeholder object in the object properties in the mapping and then specify an override to this placeholder object from the advanced properties. Ensure that the placeholder object contains the same metadata as the corresponding table that you specify as an override. When you run the mapping, the value specified in the advanced property overrides the placeholder object.

Parameterizing both the object and the advanced properties

If you want to keep both the Snowflake object type and the advanced fields parameterized, you must leave the **Allow parameter to be overridden at runtime** option unselected in the input parameter window while adding the parameters, and then select the required object at the task level. When you run the task, the values specified in the advanced properties take precedence.

Parameterization is not applicable for mappings that use the **Create Target** option. In advanced mode, only input parameters are applicable for migration.

Migration restrictions

You need to consider some rules for some of the configurations before you migrate assets.

Migrate mappings containing multiple source objects

If you want to override the database and schema in the advanced properties for source objects, the database and schema selected in all the source objects must be the same.

For example, a mapping contains the following objects:

object1: `<db1>.<schema1.<object1>`

object2: `<db1>.<schema1.object2`

For example, the following image shows the two source objects used in the mapping where the database and schema name for both the sources are the same:
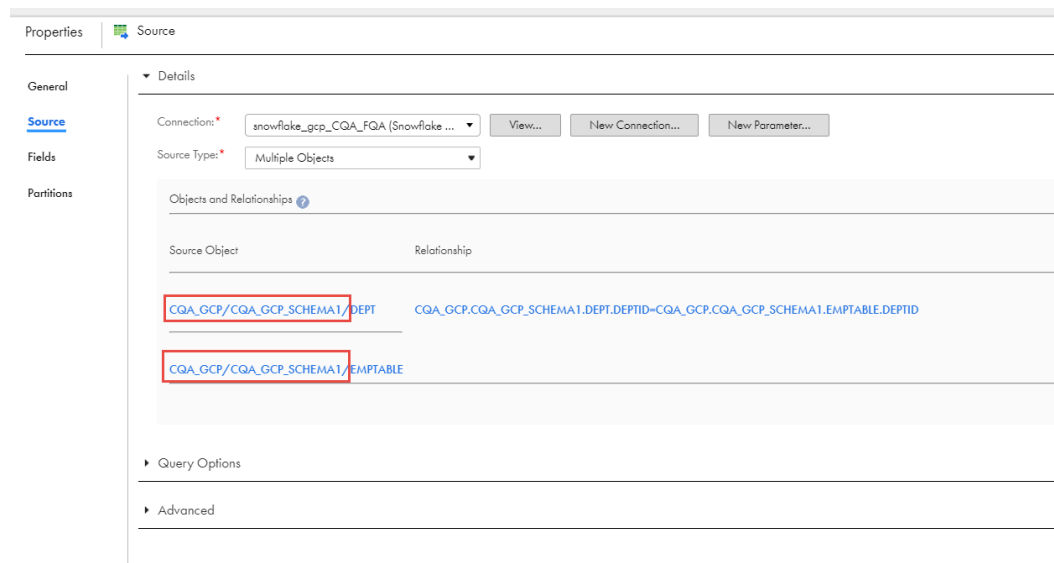
If the database and schema are different, for example, if the database and schema for object1 is `<db1>.<schema1.object1>` and object2 is `<db2>.<schema2.object2>`, migration does not work.

When the mapping contains multiple objects, do not override the table from the advanced properties.

Migrate mappings containing advanced filter and a table name override

If the mapping contains an advanced filter for the object, for example, <Table2>, in the query options section and an override to the table name in the advanced properties, consider the following rules to override the table:

In mappings, consider one of the following options:

- The advanced filter in the mapping must include the *$$Filtercondition* value. In the mapping task, you can override the value from the parameter file in the task properties. The parameter condition in the parameter file must include the following condition: `$$Filtercondition =<Table2>.id >= 1`

- Use the table name that you specify as an override in the advanced properties directly in the advanced filter condition. For example, `<Table2>.id >= 1`.

In advanced mode, use the table name specified as an override in the object advanced properties directly in the advanced filter condition. For example, `<Table2>.id >= 1`.

Migrate mappings containing an SQL override and custom query

In the mapping, when you specify an SQL override and the object type used is a custom query, ensure that the table you specified in the custom query and the SQL override contains the same metadata. The custom query and SQL override must be fully qualified. For example, `Select * from DB1.Schema.Table1`

When the mapping contains an SQL override and custom query combination, do not specify the database or schema in the advanced source properties in the mapping and in the Additional JDBC URL Parameters field in the Snowflake connection If you do, the mapping fails.

When you configure a partially parameterized custom query, specify the default value for the parameters. For example, when you specify `Select * from $$DB1.$$Schema.$$Table1`, add a placeholder value as the default value while creating the in-out parameters for DB1, Schema1, and Table1.

If you have selected parameter as the source type in the mapping, do not select the **Allow parameter to be overridden at runtime** option. You can then select the custom query in the mapping task.

For more information about the supported overrides in combination with an SQL override from the advanced properties, see .

Part III: SQL ELT with Snowflake Data Cloud Connector

This part contains the following chapters:

CHAPTER 8

Introduction to SQL ELT

You can enhance the mapping performance with SQL ELT.

You can read data from a cloud data warehouse and write it to the same cloud data warehouse. You can also read data from a data lake in your cloud ecosystem and write it to a cloud data warehouse in the same ecosystem. Data Integration translates the transformation logic into ecosystem-specific commands and SQL statements that run in the underlying cloud infrastructure. This increases the data processing speed because the data isn't moved out of the cloud infrastructure for processing.

Example

You work for healthcare solutions and your organization provides healthcare technology to pharmacies and pharmacy chains. You enable pharmacies to process prescriptions, store and provide access to healthcare records, and improve patient outcomes. Your organization stores its data in Amazon S3.

The management wants to create a patient-centric pharmacy management system. The organization plans to leverage the warehouse infrastructure of Snowflake and load all its data to Snowflake so that they can make operational, financial, and clinical decisions with ease.

To load data from an Amazon S3 object to Snowflake, you must use SQL ELT with the required transformations that support the data warehouse model. Use an Amazon S3 V2 connection to read data from the Amazon S3 source and a Snowflake Data Cloud connection to write to a Snowflake target. The Amazon S3 source data is uploaded to the Snowflake stage using the PUT command. The Snowflake COPY commands are used to convert the transformations to the corresponding SQL functions and expressions while loading the data to Snowflake. You can enhance the performance of the task and reduce the costs involved by configuring SQL mappings.

Configuring SQL ELT

You can configure SQL ELT when you want the mapping logic to be processed by your cloud ecosystem.

Configure SQL ELT in one of the following ways:

Create a mapping in SQL ELT mode

In a mapping in SQL ELT mode, you don't have to specify an optimization type. Data Integration pushes the transformation logic by default to the target database.

You can also preview data for individual transformations to validate the mapping logic.

When you run a mapping in SQL ELT mode, you can load data from the following data sources to Snowflake:

- Amazon S3

- Microsoft Azure Data Lake Storage Gen2
- Snowflake

For more information on mappings in SQL ELT mode, see Chapter 10, "Mappings in SQL ELT mode for Snowflake Data Cloud" on page 76.

Create a mapping or mapping in advanced mode

If you want to push some or all the transformation logic based on the selected optimization type, you can add the mapping or mapping in advanced mode to a mapping task, and enable SQL ELT optimization in the mapping task. Data Integration pushes the transformation logic to the source or target database based on the optimization type you specify in the task properties. On the **Runtime Options** tab, you can configure Full, Source, and other optimization configurations. Data Integration processes any transformation logic that isn't pushed to the sources and targets.

When you create mappings and enable SQL ELT optimization in a mapping task, you can load data from the following data sources to Snowflake:

- Amazon S3
- Google Cloud Storage
- Microsoft Azure Data Lake Storage Gen2
- Snowflake

For more information on SQL ELT optimization, see Chapter 11, "SQL ELT optimization for mapping tasks" on page 89.

Previewing SQL ELT query

Before you run a mapping that is configured for SQL ELT optimization or a mapping in SQL ELT mode, you can preview SQL queries in the SQL ELT Query panel in the Mapping Designer.

After you configure a mapping and run the preview, Data Integration creates and runs a temporary SQL ELT preview mapping task. When the job completes, Data Integration displays the SQL queries to be executed and any warnings in the **SQL ELT Query** panel. The warning messages help you understand which transformations in the configured mapping are not applicable for SQL ELT optimization. If SQL ELT query preview fails, Data Integration lists any queries generated up to the point of failure. You can edit the mapping and fix the required transformations before you run the mapping for SQL ELT optimization.

You can also view the temporary job created under **My Jobs** and download the session log to view the queries generated.

For more information about how to preview SQL ELT query, see "SQL ELT query preview" in *Mappings* in the Data Integration documentation.

CHAPTER 9

Prepare for SQL ELT

Before you can configure SQL ELT to load data from different sources to Snowflake, you need to meet certain requirements:

- Access to data files in a Google Cloud Storage bucket
- Access to data files in a Microsoft Azure Data Lake Storage Gen2 container
- Verify permissions to run mappings in SQL ELT mode

Access to data files in a Google Cloud Storage bucket

You must have access to data files in a Google Cloud Storage bucket to load data from Google Cloud Storage to Snowflake.

To do this, perform the following steps:

1. Create a Cloud Storage Integration object that contains the details of the Google Cloud Storage buckets from which you want to read data.
2. After you create the Cloud Storage Integration in Snowflake, specify the Cloud Storage Integration name in the **Additional JDBC URL Parameters** connection property.
   The Storage Integration value is case-sensitive.

Snowflake Data Cloud Connector creates a temporary external stage that uses the Cloud Storage Integration you created.

Configuring storage integration for Google Cloud Storage

Create a storage integration to allow Snowflake to read data from the Google Cloud Storage bucket.

You can refer to the Snowflake documentation to perform the following steps:

1. Create a Cloud Storage Integration in Snowflake.
2. Retrieve the Cloud Storage Service account for your Snowflake account.
3. Grant the service account permissions to access the bucket objects.
   a. Create a custom IAM role.
   b. Assign the custom role to the Cloud Storage Service account.
4. Grant permissions for the role to create an external stage.

The role must have the CREATE STAGE privilege on the schema and the USAGE privilege on the storage integration.

For example, run the following commands to grant these privileges:

```
grant create stage on schema public to role myrole;
grant usage on integration gcs_int to role myrole;
```

Access to data files in a Microsoft Azure Data Lake Storage Gen2 container

You must have access to data files in a Microsoft Azure Data Lake Storage Gen2 container to load data from Microsoft Azure Data Lake Storage Gen2 to Snowflake.

To do this, perform the following steps:

1. Create a Cloud Storage Integration object that contains the details of the Microsoft Azure Data Lake Storage Gen2 container from which you want to read data.

2. After you create the Cloud Storage Integration in Snowflake, specify the Cloud Storage Integration name in the **Additional JDBC URL Parameters** connection property.
   The Storage Integration value is case-sensitive.

The Snowflake Data Cloud Connector creates a temporary external stage that uses the Cloud Storage Integration you created.

Configuring storage integration for Microsoft Azure Data Lake Storage Gen2

Create a storage integration to allow Snowflake to read data from the Microsoft Azure Data Lake Storage Gen2 container.

You can refer to the [Snowflake](#) documentation to perform the following steps:

1. Create a Cloud Storage Integration in Snowflake.

2. Retrieve the Cloud Storage Service account for your Snowflake account. See

3. Grant the service account permissions to access the bucket objects.

   a. Create a custom IAM role.

   b. Assign the custom role to the Cloud Storage Service account.

4. Grant permissions for the role to create an external stage.

   The role must have the CREATE STAGE privilege on the schema and the USAGE privilege on the storage integration.

   For example, run the following commands to grant these privileges:

```
grant create stage on schema public to role myrole;
grant usage on integration adls_int to role myrole;
```

Granting access to the storage locations

Grant the Snowflake service principal access to the Azure Services storage accounts.

1. Run the DESCRIBE INTEGRATION command to retrieve the following consent URL: `desc storage integration <integration_name>;`

   where integration_name is the name of the integration you created.
   The URL in the AZURE_CONSENT_URL column has the following format:

   ```
   https://login.microsoftonline.com/<tenant_id>/oauth2/authorize?
   client_id=<snowflake_application_id
   ```

   Copy the value in the AZURE_MULTI_TENANT_APP_NAME column. This is the name of the Snowflake client application created for your account. You need this information to grant this application the required permissions to get an access token for the storage locations.

2. In a web browser, navigate to the URL in the AZURE_CONSENT_URL URL column.

   The page displays a Microsoft permissions request page.

3. Click **Accept**.

   This allows the Azure service principal created for your Snowflake account to obtain an access token on any resource inside your tenant. The access token is generated successfully only if you grant the service principal the appropriate permissions on the container.

4. Log into the Microsoft Azure portal.

5. Navigate to **Azure Services** > **Storage Accounts**, and then click the name of the storage account for which you want to grant the Snowflake service principal access to.

6. Click **Access Control (IAM)** > **Add Role Assignment**.

7. Select the required role to grant to the Snowflake service principal:

   - Storage Blob Data Reader: Grants read access only. You can load data from files staged in the storage account.

   - Storage Blob Data Contributor: Grants read and write access. You can load data from or unload data to files staged in the storage account.

8. Search for the Snowflake service principal.

   This is the identity in the AZURE_MULTI_TENANT_APP_NAME property in the DESC STORAGE INTEGRATION output in Step 1. It might take an hour or longer for Azure to create the Snowflake service principal requested through the Microsoft request page. If the service principal is not available immediately, it is recommended that you wait for an hour or two and then search again. If you delete the service principal, the storage integration stops working.

9. Click **Save**.

   The role assignments might take up to five minutes to take affect.

Verify permissions for SQL ELT

Permissions define the level of access for the operations that you can perform in Snowflake.

The following table lists the permissions that you require in the Snowflake account to run mappings in SQL ELT mode:

Object Type	Privileges
Database	Usage
Schema	Usage, Create Table, Create View, Create Sequence, Create Function, Create Stage
Table	All
Sequence	All
Stored Procedure	All
User Defined Function (required for semi-structured data types)	All
View	All

For more information, see Access Control Privileges in the Snowflake documentation.

CHAPTER 10

Mappings in SQL ELT mode for Snowflake Data Cloud

You can create mappings in SQL ELT mode to read data from your Snowflake cloud data warehouse or from Snowflake, Amazon S3, or Microsoft Azure Data Lake Storage Gen2 in your cloud ecosystem, load it to your Snowflake cloud data warehouse, and perform all of the data transformation within Snowflake.

To create a mapping in SQL ELT mode, you create a mapping, and then select **Mapping - SQL ELT** as the mapping type. You're then prompted to choose a Snowflake target connection. If your organization doesn't have any Snowflake connections, you're prompted to create one. Before you create mappings in SQL ELT mode, be sure to complete the prerequisites. For more information, see Chapter 9, "Prepare for SQL ELT" on page 72.

After you choose the target connection, the Mapping Designer opens. When you create a mapping, a Source transformation and a Target transformation are already on the canvas for you to configure.

Sources in mappings in SQL ELT mode

When you configure the source connection in the Source transformation, you can choose only an Amazon S3 V2, Microsoft Azure Data Lake Storage Gen2, or a Snowflake Data Cloud connection.

Snowflake Data Cloud source properties

You can configure the following properties for a Snowflake Data Cloud source:

- Source connection - Parameter, Allow parameter to be overridden at run time
- Source type - Single object, multiple objects, query, and parameter. You can also use a parameter file to override the Snowflake source connections and objects in a mapping from the mapping task.
- Allow parameter to be overridden at run time.
- Query options - Filter and Join. You can use both simple and advanced filter conditions. You can use join to join related objects based on existing relationships or you can create an advanced relationship.
- Database
- Schema
- Table name
- SQL override

Amazon S3 V2 source properties

You can configure the following properties for an Amazon S3 V2 source:

- Source connection parameter
- Source Type - Single, parameter
- Format - Delimited, Avro, ORC, Parquet, and JSON
- Source Type - File and directory
- Folder Path
- File Name
- Compression Format - Gzip

For information on how to configure the supported properties, see the Amazon S3 V2 Connector documentation.

Microsoft Azure Data Lake Storage Gen2 source properties

You can configure the following properties for a Microsoft Azure Data Lake Storage Gen2 source:

- Source connection, connection parameter
- Source Type - Single, parameter
- Format - Delimited, Avro, ORC, Parquet, and JSON
- Formatting Options
- Filesystem Name Override
- Source Type - File and directory
- Directory Override - Absolute path; Relative path
- File Name Override - Source object
- Compression Format - Gzip to read flat files

For information on how to configure the supported properties, see the Microsoft Azure Data Lake Storage Gen2 Connector documentation.

For information on how the sources in mappings in SQL ELT mode behave differently from the sources in other types of mappings, see Sources in mappings in SQL ELT mode in *Mappings* in the Data Integration help.

Targets in mappings in SQL ELT mode

When you configure a Target transformation in a mapping in SQL ELT mode, you need to use only a Snowflake Data Cloud connection.

You can add multiple targets to the mapping and view statistics for each target separately in the job details page, but all targets must use the same target connection or connection parameter.

You can configure the following properties in a Snowflake target transformation in a mapping in SQL ELT mode:

- Target connection - Parameter, Allow parameter to be overridden at run time.
- Target type - Single object, parameter.
- Allow parameter to be overridden at run time.

- Target object - Existing target, Create new at runtime.

- Operation - Insert, update, delete, or data driven.

- Update Columns

- Update Mode

- Database

- Schema

- Pre-SQL and Post-SQL

- Truncate Target Table

- Additional Write Runtime Parameters

- Table name

The following target properties don't apply in a mapping in SQL ELT mode:

- Warehouse

- Role

For information on how the targets in mappings in SQL ELT mode behaves differently from the targets in other types of mappings, see Targets in mappings in SQL ELT mode in *Mappings* in the Data Integration help.

Transformations in mappings in SQL ELT mode

A mapping in SQL ELT mode includes transformations that Snowflake Data Cloud can process.

You can use the following transformations in a mapping in SQL ELT mode:

- Source

- Target

- Aggregator

- Expression

- Filter

- Joiner

- Lookup

- Normalizer

- Rank

- Router

- Sequence Generator

- Sorter

- Union

Functions in mappings in SQL ELT mode

When you create expressions within a mapping in SQL ELT mode, you must use the native functions and expression syntax of Snowflake Data Cloud and not Informatica functions and expression syntax.

You can use the following native functions in a mapping in SQL ELT mode:

Aggregate functions

Function	Function	Function
ANY_VALUE(expr)	COVAR_SAMP(y, x)	REGR_INTERCEPT(y,x)
APPROX_COUNT_DISTINCT(expr)	HASH_AGG(expr)	REGR_R2(y,x)
APPROX_PERCENTILE_ACCUMULATE(expr)	HLL(expr)	REGR_SLOPE(y,x)
ARRAY_UNIQUE_AGG(column)	HLL_ACCUMULATE(expr)	REGR_SXX(y,x)
AVG(expr)	HLL_COMBINE(state)	REGR_SXY(y,x)
BITAND_AGG(expr)	KURTOSIS(expr)	REGR_SYY(y,x)
BITOR_AGG(expr)	MAX(expr)	STDDEV(expr)
BITXOR_AGG(expr)	MAX_BY(col_to_return, col_containing_max)	STDDEV_POP(expr)
BOOLAND_AGG(expr)	MEDIAN(expr)	STDDEV_SAMP(expr)
BOOLOR_AGG(expr)	MIN(expr)	SUM(expr)
BOOLXOR_AGG(expr)	MIN_BY(col_to_return, col_containing_max)	VAR_POP(expr)
CORR(y, x)	MODE(expr)	VAR_SAMP(expr)
COUNT(expr)	REGR_AVGX(y,x)	VARIANCE(expr1)
COUNT_IF(cond)	REGR_AVGY(y,x)	VARIANCE_POP(expr)
COVAR_POP(y, x)	REGR_COUNT(y,x)	VARIANCE_SAMP(expr1)

Bitwise expression functions

Function	Function	Function
BITAND(expr1, expr2)	BITSHIFTLEFT(expr1, n)	GETBIT(integer_expr, bit_position)
BITNOT(expr)	BITSHIFTRIGHT(expr1, n)	-
BITOR(expr1, expr2)	BITXOR(expr1, expr2)	-

Conditional expression functions

Function	Function	Function
BOOLAND(expr1, expr2)	GREATEST(expr)	NVL(expr1,expr2)
BOOLNOT(expr)	IFF(condition, expr1, expr2)	NVL2(expr1,expr2,expr3)
BOOLOR(expr1, expr2)	IFNULL(expr1, expr2)	REGR_VALX(expr1,expr2)
BOOLXOR(expr1, expr2)	LEAST(expr)	REGR_VALY(expr1,expr2)
COALESCE(expr1,expr2)	NULLIF(expr1,expr2)	ZEROIFNULL(expr)
EQUAL_NULL(expr1,expr2)	NULLIFZERO(expr)	-

Context functions

Function	Function	Function
ALL_USER_NAMES()	CURRENT_SCHEMA()	INVOKER_ROLE()
CURRENT_ACCOUNT()	CURRENT_SCHEMAS()	INVOKER_SHARE()
CURRENT_AVAILABLE_ROLES()	CURRENT_SECONDARY_ROLES()	IS_GRANTED_TO_INVOKER_ROLE(role_name)
CURRENT_CLIENT()	CURRENT_SESSION()	IS_ROLE_IN_SESSION(role_name)
CURRENT_DATABASE()	CURRENT_STATEMENT()	LAST_QUERY_ID(num)
CURRENT_DATE()	CURRENT_TIMESTAMP(fract_sec_precision)	LAST_TRANSACTION()
CURRENT_IP_ADDRESS()	CURRENT_TRANSACTION()	LOCALTIMESTAMP()
CURRENT_REGION()	CURRENT_USER()	SYSDATE()
CURRENT_ROLE()	CURRENT_VERSION()	-
CURRENT_ROLE_TYPE()	CURRENT_WAREHOUSE()	-

Conversion functions

Function	Function	Function
DATE(String)	TO_NUMBER(Boolean)	TO_VARCHAR(Binary)
DATE(Temporal)	TO_NUMBER(String)	TO_VARCHAR(Numeric)
TO_BINARY	TO_NUMERIC	TO_VARCHAR(Temporal)
TO_BINARY(String)	TO_NUMERIC(Boolean)	TRY_TO_BINARY

Function	Function	Function
TO_BOOLEAN	TO_NUMERIC(String)	TRY_TO_BOOLEAN
TO_BOOLEAN(Number)	TO_TIMESTAMP	TRY_TO_DATE
TO_CHAR	TO_TIMESTAMP(String)	TRY_TO_DECIMAL
TO_CHAR(Binary)	TO_TIMESTAMP(Temporal)	TRY_TO_DOUBLE
TO_CHAR(Numeric)	TO_TIMESTAMP_LTZ	TRY_TO_NUMBER
TO_CHAR(Temporal)	TO_TIMESTAMP_LTZ(String)	TRY_TO_NUMERIC
TO_DECIMAL(Boolean)	TO_TIMESTAMP_LTZ(Temporal)	TRY_TO_TIMESTAMP
TO_DECIMAL(String)	TO_TIMESTAMP_NTZ	TRY_TO_TIMESTAMP_LTZ
TO_DOUBLE	TO_TIMESTAMP_NTZ(String)	TRY_TO_TIMESTAMP_NTZ
TO_DOUBLE(String)	TO_TIMESTAMP_NTZ(Temporal)	-
TO_NUMBER	TO_VARCHAR	-

Cortex functions

Function
SNOWFLAKE_CORTEX_COMPLETE(model, prompt_or_history)
SNOWFLAKE_CORTEX_EXTRACT_ANSWER(source_document, question)
SNOWFLAKE_CORTEX_SENTIMENT(text)
SNOWFLAKE_CORTEX_SUMMARIZE(text)
SNOWFLAKE_CORTEX_TRANSLATE(text, source_language, target_language)

Data generation functions

Function	Function	Function
NORMAL(mean,stddev,gen)	SEQ2([0|1])	UUID_STRING(uuid,name)
RANDOM()	SEQ4([0|1])	ZIPF(s, N, gen)
RANDSTR(length,gen)	SEQ8([0|1])	-
SEQ1([0|1])	UNIFORM(min,max,gen)	-

Date and time functions

Function	Function
ADD_MONTHS(date_or_time_expr, num_months_expr)	NEXT_DAY(date_or_time_expr, dow_string)
DATEADD(date_or_time_part, value, date_or_time_expr)	PREVIOUS_DAY(date_or_time_expr, dow_string)
DATEDIFF(date_or_time_part, date_or_time_expr1, date_or_time_expr2)	QUARTER(date_or_timestamp_expr)
DATE_FROM_PARTS(year, month, day) or DATEFROMPARTS(year, month, day)	SECOND(date_or_timestamp_expr)
DATE_PART(date_or_time_part,date_or_time_expr)	SYSDATE()
DATE_TRUNC(date_or_time_part,date_or_time_expr)	TIMESTAMPADD(date_or_time_part,value,date_or_time_expr)
DAY(date_or_timestamp_expr)	TIMESTAMPDIFF(date_or_time_part,date_or_time_expr1,date_or_time_expr2)
DAYNAME(date_or_timestamp_expr)	TIMESTAMP_LTZ_FROM_PARTS(year, month, day, hour, minute, second) or TIMESTAMPLTZFROMPARTS(year, month, day,hour, minute, second)
DAYOFMONTH(date_or_timestamp_expr)	TIMESTAMP_NTZ_FROM_PARTS(year, month, day, hour, minute, second) or TIMESTAMPNTZFROMPARTS(year, month, day, hour, minute, second)
DAYOFWEEK(date_or_timestamp_expr)	TIME_SLICE(date_or_time_expr, slice_length,date_or_time_part)
DAYOFWEEKISO(date_or_timestamp_expr)	TRUNC(date_or_time_expr, date_or_time_part)
DAYOFYEAR(date_or_timestamp_expr)	WEEK(date_or_timestamp_expr)
HOUR(date_or_timestamp_expr)	WEEKISO(date_or_timestamp_expr)
LAST_DAY(date_or_time_expr)	WEEKOFYEAR(date_or_timestamp_expr)
MINUTE(date_or_timestamp_expr)	YEAR(date_or_timestamp_expr)
MONTH(date_or_timestamp_expr)	YEAROFWEEK(date_or_timestamp_expr)
MONTHNAME(date_or_time_expr)	YEAROFWEEKISO(date_or_timestamp_expr)
MONTHS_BETWEEN(date_expr1, date_expr2)	-

Encryption functions

Function
DECRYPT(value_to_decrypt,passphrase)
DECRYPT_RAW(value_to_encrypt,key,iv)
ENCRYPT(value_to_encrypt,passphrase)
ENCRYPT_RAW(value_to_encrypt,key,iv)

Numeric functions

Function	Function	Function
ABS(expr)	DEGREES(real_expr)	ROUND(input_expr)
ACOS(real_expr)	DIV0(dividend,divisor)	SIGN(expr)
ACOSH(real_expr)	DIV0NULL(dividend,divisor)	SIN(real_expr)
ASIN(real_expr)	EXP(real_expr)	SINH(real_expr)
ASINH(real_expr)	FACTORIAL(integer_expr)	SQRT(expr)
ATAN(real_expr)	FLOOR(expr)	SQUARE(expr)
ATAN2(y,x)	LN(expr)	TAN(real_expr)
ATANH(real_expr)	LOG(base,expr)	TANH(real_expr)
CBRT(expr)	MOD(expr1,expr2)	TRUNCATE(expr)
CEIL(expr)	PI(real_expr)	TRUNC(expr)
COS(real_expr)	POW(x,y)	WIDTH_BUCKET(expr,min_value,max_value,num_buckets)
COSH(real_expr)	POWER(x,y)	-
COT(real_expr)	RADIANS(real_expr)	-

Regular expression functions

Function
REGEXP_COUNT(subject,pattern)
REGEXP_INSTR(subject,pattern)
REGEXP_LIKE(subject,pattern)
REGEXP_REPLACE(subject,pattern)

Function
REGEXP_SUBSTR(subject,pattern)
REGEXP_SUBSTR_ALL(subject,pattern)
RLIKE(subject,pattern)

String and binary functions

Function	Function	Function
ASCII(expr)	JAROWINKLER_SIMILARITY(string_expr1,string_expr2)	RTRIMMED_LENGTH(expr)
BASE64_DECODE_BINARY(input [, alphabet])	LEFT(string_expr,length_expr)	SHA1(expr)
BASE64_DECODE_STRING(input [, alphabet])	LEN(expr)	SHA1_HEX(expr)
BASE64_ENCODE(input)	LENGTH(expr)	SHA1_BINARY(expr)
BIT_LENGTH(expr)	LIKE(subject, pattern)	SHA2_HEX(expr)
BIT_LENGTH(expr)	LOWER(expr)	SHA2(expr)
CHR(expr)	LPAD(expr1, expr2)	SHA2_BINARY(expr)
CHAR(expr)	LPAD(expr1, expr2, expr3)	SOUNDEX(varchar_expr)
CHARINDEX(expr1, expr2)	LTRIM(expr)	SOUNDEX_P123(varchar_expr)
COLLATE(string_expression,collation_specification)	MD5(expr)	SPACE(expr)
COLLATION(expression)	MD5_BINARY(expr)	SPLIT(expr1, expr2)
COMPRESS(input,method)	MD5_HEX(expr)	SPLIT_PART(string,delimiter,part_number)
CONCAT(expr)	MD5_NUMBER_LOWER64(msg)	STARTSWITH(expr1, expr2)
CONCAT_WS(expr1, expr2)	MD5_NUMBER_UPPER64(msg)	STRTOK(string)
CONTAINS(expr1, expr2)	OCTET_LENGTH(string)	STRTOK_TO_ARRAY(string)
DECOMPRESS_BINARY(input,method)	PARSE_IP(expr,type)	SUBSTR(base, start)
DECOMPRESS_STRING(input,method)	PARSE_URL(expr)	SUBSTRING(base, start)
EDITDISTANCE(expr1, expr2)	POSITION(expr1, expr2)	TRANSLATE(subject,sourceAlphabet,targetAlphabet)

Function	Function	Function
ENDSWITH(expr1, expr2)	REPEAT(expr1, expr2)	TRIM(expr)
HASH(expr1)	REPLACE(subject, pattern)	TRY_BASE64_DECODE_BINARY(input)
HEX_DECODE_BINARY(input)	REVERSE(expr)	TRY_BASE64_DECODE_STRING(input)
HEX_DECODE_STRING(input)	RIGHT(expr, len)	TRY_HEX_DECODE_BINARY(input)
HEX_ENCODE(input)	RIGHT(expr1, len)	TRY_HEX_DECODE_STRING(input)
ILIKE(subject,pattern)	RPAD(expr1, expr2)	UNICODE(expr)
INITCAP(expr)	RPAD(expr1, expr2, expr3)	UPPER(expr)
INSERT(expr1, pos, len, expr2)	RTRIM(expr1)	-

Window functions

Function	Function	Function
ANY_VALUE(expr)	BOOLXOR_AGG(expr)	RATIO_TO_REPORT(expr)
APPROX_COUNT_DISTINCT(expr)	CORR(y, x)	REGR_AVGX(y,x)
APPROX_PERCENTILE(expr,percentile)	COVAR_POP(y, x)	REGR_AVGY(y,x)
APPROX_PERCENTILE_ACCUMULATE(expr)	COVAR_SAMP(y, x)	REGR_COUNT(y,x)
APPROX_TOP_K(expr)	HASH_AGG(expr)	REGR_INTERCEPT(y,x)
APPROX_TOP_K_ACCUMULATE(expr,counters)	HLL(expr)	REGR_R2(y,x)
ARRAY_UNIQUE_AGG(column)	HLL_ACCUMULATE(expr)	REGR_SLOPE(y,x)
BITAND_AGG(expr)	HLL_COMBINE(state)	REGR_SXX(y,x)
BITOR_AGG(expr)	KURTOSIS(expr)	REGR_SXY(y,x)
BITXOR_AGG(expr)	KURTOSIS(expr)	REGR_SYY(y,x)
BOOLAND_AGG(expr)	MINHASH(k,expr)	-
BOOLOR_AGG(expr)	MODE(expr)	-

For information on functions and expression syntax, see SQL function reference in the Snowflake documentation.

Operators in mappings in SQL ELT mode

When you use mappings in SQL ELT mode, Data Integration converts the expression in the transformation by determining equivalent operators in the database. If there is no equivalent operator, Data Integration processes the transformation logic.

The table lists the operators that you can push to Snowflake:

Operator	Operator
+	<>
-	>=
*	<=
/	!=
%	AND
>	OR
<	NOT
=	-

Rules and guidelines in mappings in SQL ELT mode

Consider the following rules and guidelines when you run mappings in SQL ELT mode:
General guidelines

- A mapping that uses input parameters for a custom query to read from a Snowflake table fails.
- To process a data type used in the Expression transformation, ensure that the precision of the data type matches the precision of the data type returned by Snowflake. Otherwise, the mapping fails.

Data types

- When you read from and write to Snowflake, ensure that the source data doesn't contain the Time and TIMESTAMPTZ data types.
- A mapping that reads the Array data type from Microsoft Azure Data Lake Storage Gen2 and writes to Snowflake fails, if the Array data type contains the boolean data.
- When you use input parameters for the source filter, ensure that you use only the String data type in the filter expression.
- You can't read the Array data type from the Amazon S3 V2 source in a mapping in SQL ELT mode.
- A mapping that reads the Array or Object data type from a Snowflake source and write to a Snowflake target created at run time fails.
- When you run a mapping to write data of the DATE or TIMESTAMPLTZ data type to a Snowflake target created at run time, the mapping converts the date/time data type to the TIMESTAMPNTZ data type and writes the TIMESTAMPNTZ data type with a default precision of 9.

Functions

- When you pass date or time values in the Date or Time function, the mapping fails. To run mapping successfully, select the required fields from the date and time function list to define the field expression and run the mapping.

- When you pass the date_or_time_part argument in the date or time function, ensure that you enclose the date_or_time_part argument with the single quote character.
  For example, when you use the DATEADD function, use the following format:

  ```
  DATEADD('Year', <Value>, <Field name>)
  ```

- When you use the TO_NUMERIC or TO_NUMBER function to process the boolean data type, you need to configure the field that contains the boolean data type as an argument to the function.
  For example, when you use the TO_NUMBER function, use the following syntax in the Expression transformation:

  ```
  TO_NUMBER(TO_BOOLEAN(bool_field))
  ```

- Ensure that you pass the uuid and name arguments for the UUID_STRING function.
  For example, when you use the UUID_STRING function, use the following format:

  ```
  UUID_STRING(uuid,name)
  ```

  If you pass the UUID_STRING function without any argument, Snowflake returns a random UUID.

- To configure the Window functions in an Expression transformation, you need to select the **Enable window properties** check box on the **Window** tab.

- You can't pass more than four arguments for a function in an Expression transformation.

- To pass an integral argument for the APPROX_TOP_K, APPROX_TOP_K_ACCUMULATE, or MINHASH function, add an integer constant in the argument for the integral argument.
  For more information about how to pass integral arguments, see the Snowflake documentation.

- To pass the percentile argument for the APPROX_PERCENTILE function, add a numeric constant in the argument that ranges from 0 to 1.
  For more information about how to pass percentile arguments, see the Snowflake documentation.

- To configure a function that contains multiple arguments with the ANY data type, ensure that the data type for the arguments are homogeneous.
  For example, when you pass the expr1 and expr2 arguments with the ANY data type for the NVL function and the expr1 argument contains the integer data type, the expr2 argument also needs to be of the integer data type.

Target operations

- When you define the warehouse name and role assigned to the user in the Target transformation, the mapping doesn't honor the value specified in the **Warehouse** and **Role** target properties. You need to specify the warehouse name and role in the Snowflake Data Cloud connection.

- To perform an upsert operation on the target, you need to select the target operation as **Update** and update mode as **Update Else Insert** in the Snowflake Target transformation.

Expression validations

- When you configure the SEQ1, SEQ2, SEQ4, or SEQ8 function in an expression, you can pass 0 or 1 value in the optional arguments. If you pass any other value, the expression validates the argument, but the mapping fails at run time.

- When you configure a function with an expression that returns the String data type and the output port is Integer, Bigint, Text, Decimal, or Double data type, even though the expression validates the data type, the mapping fails at run time.

- Ensure that the string argument for the Date or Time function is valid. Even when the argument is not valid, the expression validates it, but the mapping fails at run time.

- When you configure a Window function that is not listed in an Aggregator transformation, the expression validates the function, but the mapping fails at run time.

- The expression validation fails for an argument in the following scenarios:
  - If you configure the UNIFORM function with an expression that contains the cast operator (::).
  - If you configure a function with an expression that contains the DISTINCT keyword or asterisk (*) wildcard.
  - If you pass a string argument with Numeric values for the Bitwise expression function.

CHAPTER 11

SQL ELT optimization for mapping tasks

You can configure SQL ELT optimization from a mapping task to enhance the mapping performance.

After you create a mapping or a mapping in advanced mode, add the mapping to a mapping task, and then configure SQL ELT optimization in the mapping task. You can select how Data Integration handles SQL ELT optimization in the **SQL ELT Optimization Fallback Option** menu on the **Runtime Options** tab.

The task converts the transformation logic to Snowflake queries, sends the queries to Snowflake, and the mapping logic is processed in the Snowflake database.

If your mapping contains multiple pipelines, you can define the flow run order to load the targets from the pipelines in a particular order.

You can configure SQL ELT optimization for a mapping or a mapping in advanced mode in the following scenarios:

Snowflake to Snowflake

Read from and write to Snowflake using a Snowflake Data Cloud connection.

Amazon S3 to Snowflake

Read from Amazon S3 using an Amazon S3 V2 connection in the Source transformation and write to Snowflake using a Snowflake Data Cloud connection in the Target transformation.

Microsoft Azure Data Lake Storage Gen2 to Snowflake

Read from Microsoft Azure Data Lake Storage Gen2 using a Microsoft Azure Data Lake Storage Gen2 connection in the Source transformation and write to Snowflake using a Snowflake Data Cloud connection in the Target transformation.

Google Cloud Storage to Snowflake

Read from Google Cloud Storage using a Google Cloud Storage connection in the Source transformation and write to Snowflake using a Snowflake Data Cloud connection in the Target transformation.

Note: Doesn't apply to mappings in advanced mode.

SQL ELT optimization types

When you apply SQL ELT optimization to a mapping task, Data Integration pushes the transformation logic to the source or target database based on the optimization type you specify in the task properties. Data

Integration translates the transformation logic into SQL queries or Snowflake commands to the Snowflake database based on the connection you selected.

The source or target database executes the SQL queries or Snowflake commands to process the transformations. The amount of transformation logic you can push to the database depends on the database, transformation logic, and mapping and session configuration. Data Integration processes all the transformation logic that it cannot push to a database.

You can configure the following types of SQL ELT optimization in mappings and mappings in advanced mode:

Full

Data Integration pushes down as much transformation logic as possible to process in the target database.

For mappings that read from and write to Snowflake, Data Integration analyses all the transformations from the source to the target. If all the transformations are compatible in the target, it pushes the entire mapping logic to the target. If it cannot push the entire mapping logic to the target, Data Integration first pushes as much transformation logic to the source database and then pushes as much transformation logic as possible to the target database.

Note: If the source and target Snowflake accounts are separate and reside in different regions but are hosted on the same cloud platform, you can configure full SQL ELT optimization. However, ensure that the Snowflake account user and role of the target Snowflake account has access to the Snowflake source account.

Source

Data Integration pushes down as much as the transformation logic as possible to process in the source database.

SQL ELT compatibility

You can configure the task to push transformations, variables, functions, and operators to the database.

When you use SQL ELT optimization, the Secure Agent converts the expression in the transformation by determining equivalent operators, variables, and functions in the database. If there is no equivalent operator, variable, and function, the Secure Agent processes the transformation logic.

Functions with Snowflake Data Cloud

When you use SQL ELT optimization, Data Integration converts the expression in the transformation by determining equivalent functions in the database. If there is no equivalent function, Data Integration processes the transformation logic.

The tables summarizes the availability of SQL ELT functions that you can push to Snowflake using full SQL ELT optimization:

Function	Function	Function
ABS()	IS_SPACES	SIN()
ASCII()	LAST_DAY()	SINH()

Function	Function	Function
ADD_TO_DATE()	LENGTH()	SQRT()
AVG()	LN()	STDDEV()*
CEIL()	LOG()	SUBSTR()
CHR()	LOWER()	SUM()
CONCAT()	LPAD()	SYSDATE()
COS()	LTRIM()	SYSTIMESTAMP()
COSH()	MAX()	TAN()
COUNT()	MAKE_DATE_TIME	TANH()
DATE_COMPARE()	MEDIAN()	TO_BIGINT
DATE_DIFF()	MIN()	TO_CHAR(DATE)
DECODE()	MOD()	TO_CHAR(NUMBER)
EXP()	POWER()	TO_DATE()
FLOOR()	REG_EXTRACT()	TO_DECIMAL()
GET_DATE_PART()	REG_MATCH()	TO_FLOAT()
IIF()	REG_REPLACE	TO_INTEGER()
IN()	REPLACECHR()	TRUNC(DATE)
INITCAP()	REPLACESTR()	TRUNC(NUMBER)
INSTR()	ROUND(NUMBER)	UPPER()
IS_DATE	RPAD()	MD5()
IS_NUMBER	RTRIM()	VARIANCE()
ISNULL()	SIGN()	-
*You cannot pass the filter condition argument in the STDDEV() function.		

Note: If you specify a function that is not supported for Snowflake full SQL ELT optimization, the task runs either with partial SQL ELT optimization or without full SQL ELT optimization.

Operators with Snowflake Data Cloud

When you use SQL ELT optimization, Data Integration converts the expression in the transformation by determining equivalent operators in the database. If there is no equivalent operator, Data Integration processes the transformation logic.

The tables lists the operators that you can push to Snowflake:

Operator	Operator
+	>=
-	<=
*	!=
/	AND
%	OR
||	NOT
>	IS NULL
<	IS NOT NULL
=	

Variables with Snowflake Data Cloud

You can use full SQL ELT to push the SESSSTARTTIME variable to the Snowflake database.

Transformations with Snowflake Data Cloud

When you configure SQL ELT optimization, Data Integration tries to push the configured transformation to Snowflake.

You can use full SQL ELT to push the following transformations to Snowflake:

- Aggregator
- Expression
- Filter
- Hierarchy Processor
- Joiner
- Lookup
- Normalizer
- Rank
- Router

- Sequence Generator

- SQL*

- Sorter

- Union

- Update Strategy*

*Doesn't apply to mappings in advanced mode.

Note: Router transformation is applicable only for source SQL ELT and Hierarchy Processor transformation is applicable only to mappings in advanced mode.

For more information about configuring transformations, see *Transformations* in the Data Integration documentation.

Aggregator transformation

You can configure full SQL ELT optimization to push an Aggregator transformation to process in Snowflake.

Aggregate calculations

You can perform the following aggregate calculations:

- AVG

- COUNT

- MAX

- MIN

- MEDIAN

- SUM

- VARIANCE

Incoming ports

When you configure an Aggregator transformation and the incoming port is not used in an aggregate function or in a group by field in mappings, the ANY_VALUE() function is used for columns that are not part of the group by or the aggregator function. In this case, the output is not deterministic as the ANY_VALUE() function returns any value from the port. However, in advanced mode, when the incoming port is not a part of the group by field, the MAX() function is used. The Aggregator transformation also generates an additional column with value as 1. However, this column is dropped and not used in the insert part of the SQL ELT query.

Expression transformation

You can configure full SQL ELT optimization to push an Expression transformation to process in Snowflake.

You can add an Expression transformation to each of the sources in the mapping, followed by a join downstream in the mapping. Additionally, you can add multiple Expression transformations that branch out from a transformation and then branch in into a transformation downstream in the mapping.

When you configure an Expression transformation, consider the following rules to include variables in the expression:

- You cannot use variables where you are using the value assigned while processing a previous row for calculations in the current row. if you do, the mapping runs without SQL ELT optimization.

- The variables can be nested, but you cannot refer to a variable before it is defined in the expression.

If the variables are not defined in that order, the mapping runs without SQL ELT optimization.

For example,

```
   var: AGEPLUS2 = AGEPLUS1 + 1
 var: AGEPLUS1 = AGE + 1
 out: NEXTAGE = AGEPLUS2 + 1
```

Here, AGE +1 is defined later. AGEPLUS2 in the first variable refers to AGEPLUS1 and remains unresolved.
To resolve this, specify the variables in the following order:

```
   var: AGEPLUS1 = AGE + 1
 var: AGEPLUS2 = AGEPLUS1 + 1
 out: NEXTAGE = AGEPLUS2 + 1
```

- The variables cannot have an expression that is cyclic or refers to itself:
  For example,

```
   var: AGEPLUS1 = AGEPLUS2 + 1
 var: AGEPLUS2 = AGEPLUS1 + 1
 out: NEXTAGE= AGEPLUS2
```

Here, AGEPLUS1 refers to AGEPLUS2 and remains unresolved.

Hierarchy Processor transformation

In advanced mode, you can configure a Hierarchy Processor transformation to read hierarchical or relational input from the Amazon S3 V2 or Microsoft Azure Data Lake Storage Gen2 source and write as relational or hierarchical output to the Snowflake target.

The Hierarchy Processor transformation processes hierarchical fields that represent a struct or an array.

You can configure a Hierarchy Processor transformation with the following restrictions:

- If the array element of the hierarchical output field contains the fixed-point Number data type, the mapping runs without SQL ELT optimization.

- Decimal values of the Double data type are written in exponential notation.
  For example, when you write a Double data type 2341.6789 to an output field in the Snowflake target, the output appears as 2.341678900000000e+03.

- If you select **Use input group or incoming fields** as a data source and read hierarchical or relational input from the source that contains more than one row and write as hierarchical output to a Snowflake target, Data Integration duplicates records for each row.
  To avoid writing duplicate rows to the target, either select **Inherit parent's data sources** as a data source or filter child fields from the data source using the filter condition.

- To write Integer or Bigint data type from a Struct field to a Snowflake target, select **advanced.custom.property** from the **Session Property Name** list, and then enter the following value in the mapping task:
  `DisableAdvancedMappingRuntimeValidation=true`

- You can't write data from an Avro or Parquet file that contains multi-level struct fields to a Snowflake target.

Lookup transformation

You can configure full SQL ELT optimization to push a Lookup transformation to process in Snowflake. You can push both a connected and an unconnected lookup.

When the mapping contains an unconnected lookup, you can also nest the unconnected lookup function with other expression functions. For example, `:LKP.U_LOOKUP(Upper(argument1), argument)`

Lookup objects

Consider the following rules when you configure lookups:

- You can configure a lookup for Snowflake when the Source transformation uses the following sources:
  - Amazon S3
  - Google Cloud Storage*
  - Microsoft Azure Data Lake Storage Gen2
  - Snowflake source

  *Doesn't apply to mappings in advanced mode.

- You can configure a lookup for an Amazon S3, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, or Snowflake object only when the Source transformation uses the corresponding Amazon S3, Google Cloud Storage, Microsoft Azure Data Lake Storage Gen2, or Snowflake source.

Connected lookup

In a mapping with a Snowflake Data Cloud source and target, set the **Multiple Matches** option for the lookup object to **Return all rows**.

Unconnected lookup

When you configure an unconnected Lookup transformation, consider the following rules:

- Do not configure an expression for an output received from an unconnected lookup.
- In a mapping with a Snowflake Data Cloud source and target where some of input fields from the unconnected lookup transformation is not mapped to the Snowflake target object, the select query includes all the unmapped fields.

Multiple matches behavior in a connected and unconnected lookup

If you enable SQL ELT optimization for a mapping that contains a connected or unconnected lookup, you must follow these guidelines:

- In an unconnected lookup, ensure that you always select the **Multiple Matches** option to **Report Error**. When you look up data and the lookup condition finds multiple matches, all the matching rows are selected and the task runs with SQL ELT optimization. If you enabled **Multiple Matches** to any option other than **Report Error**, the mapping runs without SQL ELT optimization.
- In a connected lookup, you can set the **Multiple Matches** option to **Return all rows** or **Report Error**. When you set the **Multiple Matches** option to **Report Error**, you can set the *Lkp_apdo_allow_report_error* custom flag in the task advanced session properties to determine how Data Integration handles multiple matches:
  - When you set the property to *Yes* and if there are multiple matches in the data, the multiple match policy is ignored and the job runs successfully with SQL ELT optimization.
  - When you do not set the property, and if there are multiple matches in the data, Data Integration considers the policy and displays a warning message. SQL ELT optimization is ignored and the task fails.

FileName port

When you configure a lookup for an Amazon S3 source in a mapping that contains an Amazon S3 source and Snowflake target, remove the filename port from both the Amazon S3 source and lookup object. The FileName port is not applicable.

Lookup query object

When you use a lookup object as a query in a Lookup transformation in a mapping to lookup data in Snowflake, specify the database and schema in the advanced lookup properties or in the additional JDBC URL parameters in the Snowflake Data Cloud connection.

Normalizer transformation

You can configure a Normalizer transformation in a mapping to return a row for each instance of the multiple-occurring data.

For example, a relational source includes four fields with quarterly sales data. You can configure a Normalizer transformation to generate a separate output row for each quarter.

In advanced mode, you can configure a Normalizer transformation with the following restrictions:

- To normalize multiple-occurring fields from different groups, map each group to a unique target and exclude all fields that are from different groups from the list of incoming fields even though the groups are not mapped to the target.

- To write normalized data to multiple targets, map all generated column IDs of normalized fields to the corresponding targets.

Router transformation

You can configure source SQL ELT optimization to push a Router transformation to the database for processing.

When you configure a Router transformation, connect or map only one output group to the target transformation.

Sequence Generator transformation

When you configure a Sequence Generator transformation, you can connect the NEXTVAL port to single or multiple ports in the transformations that follow the Sequence Generator transformation.

You can push a Sequence Generator transformation with the following restrictions:

- You cannot use a shared sequence in a Sequence Generator transformation.

- You can add only an Expression or a Target transformation after a Sequence Generator transformation in a mapping.

Guidelines for using a Sequence Generator transformation in mappings in advanced mode and SQL ELT mode

A Sequence Generator transformation in advanced mode and SQL ELT mode follows the same rules as in mappings. However, to use a Sequence Generator transformation in a mapping in advanced mode and SQL ELT mode, you need to additionally set a property in the mapping task properties, with the name of the sequence in Snowflake.

1. On the **Runtime Options** tab in the task properties, navigate to the **Advanced Session Properties** section.

2. Add a custom property and enter the following session property value in the following format:

   ```
   Pushdown.<Sequence transformation name in mapping>.SequenceName=<sequence name to
   create in Snowflake>
   ```

   For example,

   ```
   Pushdown.Seq_SF.SequenceName=snowflake_first_sequence&:Pushdown.sequence_second.Seque
   nceName=SECOND_SEQUENCE&:Pushdown.seq_third.SequenceName=THIRD_SEQ
   ```

When you use multiple Sequence Generator transformations, the name of the sequence object to be created in the Snowflake database for each sequence must be unique. If not, the transformation runs with the CREATE SEQUENCE IF NOT EXISTS query twice using the same name. The first sequence runs as expected, but the second sequence which has the same name as the first sequence does not override it. Hence, the nextValue generated for the second sequence is incorrect. However, when the sequence runs with a unique

name for the sequence object, that sequence is not updated again, irrespective of the number of times you run the mapping.

Note: If you disable SQL ELT optimization and run the mapping, the sequence values restart from the initial start value. If you delete a mapping, you must ensure to also drop the sequences from the Snowflake database.

SQL transformation

You can use an SQL transformation only to push certain functions and shared sequence.

Use functions to run queries

You can include functions in an entered query in an SQL transformation and run queries with the Snowflake target endpoint.

You must use only the SELECT clause SQL statement to push a function. Specify the column name in the select query or function. Do not push functions using statements such as "SELECT * FROM TABLE".

You can use the following functions in an entered query:

- UUID_STRING
- RANDOM
- RANDSTR
- SIGN
- CURRENT_REGION
- CURRENT_ACCOUNT
- CURRENT_ROLE
- CURRENT_USER
- CURRENT_DATABASE
- CURRENT_SCHEMA
- DAYNAME
- SPLIT
- SPLIT_PART

To use the CURRENT_ROLE, CURRENT_DATABASE, and CURRENT_SCHEMA functions in an SQL transformation, ensure to provide the database, role, and schema name in the additional JDBC parameters field in the Snowflake Data Cloud connection. If you do not specify the values in the connection, Data Integration inserts null to the target.

Reuse shared sequence

You can push a mapping with a shared sequence defined in an SQL transformation to a Snowflake endpoint. Data Integration writes the data in the same sequence to the target as in the Snowflake source.

Get the shared sequence from Snowflake and define the sequence in an entered query in an SQL transformation.

Specify the shared sequence in the entered query in the following syntax: Select `<Snowflake_schema_name>.<Snowflake_database_name>.<sequence_name>.NEXTVAL`

User defined functions

You can configure a custom query in an SQL transformation to read from Java or SQL user-defined functions (UDF) in Snowflake.

The following guidelines apply for UDFs:

- You cannot read UDFs that have newline characters in the UDF name.
- If the UDF contains array parameters, the mapping runs without SQL ELT optimization.

Union transformation

You can push a Union transformation with the following restrictions:

- The Source transformation in the mapping must only include Snowflake source objects.
- A mapping runs without SQL ELT optimization when the source is Amazon S3, Google Cloud Storage, or Microsoft Azure Data Lake Storage Gen2.

Update Strategy transformation

You cannot use an Update Strategy transformation.

You can instead use the update and upsert operations in the Target transformation to write to Snowflake.

Features

You can configure SQL ELT optimization for a mapping or mapping in advanced mode that reads from the following sources and writes to a Snowflake target:

- Snowflake source
- Amazon S3 source
- Google Cloud Storage source[1]
- Microsoft Azure Data Lake Storage Gen2 source

Note: [1]Doesn't apply to mappings in advanced mode.

When you configure a mapping, some parameters are not supported for a mapping enabled for SQL ELT optimization. You can refer to the list of parameters that each source supports.

Snowflake Data Cloud sources, targets, and lookups

When you configure SQL ELT optimization, refer to this list of supported Snowflake Data Cloud properties in the Source, Target, and Lookup transformations.

Source properties

You can configure the following properties in a Snowflake source transformation:

- Source connection - Parameter, Allow parameter to be overridden at run time
- Source type - Single object, multiple objects, query, and parameter. You can also use a parameter file to override the Snowflake source connections and objects in a mapping from the mapping task.

  Note: When you use the query source type to read from Snowflake, you can choose to retain the field metadata and save the mapping. Even if you edit the query and run the mapping, the field metadata specified at design time is retained.

- Allow parameter to be overridden at run time.

- Query options - Filter and Join. You can use both simple and advanced filter conditions. You can use join to join related objects based on existing relationships or you can create an advanced relationship.
- Database override
- Schema override
- Warehouse override
- Pre-SQL and Post-SQL
- Role override
- Table name override
- SQL override
- Tracing level

The following source properties don't apply in mappings enabled with full SQL ELT optimization:

- Warehouse override
- Pre-SQL and Post-SQL
- Role override

The following source properties don't apply in a Snowflake source transformation:

- Query options - Sort
- Partition

Target properties

You can add multiple Snowflake targets in a mapping. The target can be the same Snowflake target table added multiple times or different Snowflake target tables.

You can configure the following properties in a Snowflake target transformation:

- Target connection - Parameter, Allow parameter to be overridden at run time.
- Target type - Single object, parameter. You can also use a parameter file to override the Snowflake target connections and objects in a mapping from the mapping task.
- Allow parameter to be overridden at run time
- Target object - Existing target, Create new at runtime.
- Operation - Insert, update, upsert, delete, or data driven
- Database override
- Schema override
- Warehouse override
- Role override
- Pre-SQL and Post-SQL
- Table name override
- Truncate Target Table
- Additional Write Runtime Parameters

The following target properties don't apply in a Snowflake target transformation:

- Update Mode
- Batch row size

- Number of local staging files
- Rejected File Path
- Update Override
- Forward Rejected Rows

Note: When you write to multiple targets in mappings in advanced mode, you can only use the Insert operation.

Lookup properties

When you enable SQL ELT optimization, you can configure the following properties for Snowflake connected and unconnected lookups:

- Lookup connection - Parameter, Allow parameter to be overridden at run time
- Source type - Single object, query, parameter. You can also use a parameter file to override the Snowflake lookup connections and objects in a mapping from the mapping task.
- Multiple matches - Report Error
- Database override
- Schema override
- Warehouse override
- Role override
- Table name override
- SQL override
- Tracing level
- Lookup Data Filter

The following lookup properties don't apply in Snowflake connected and unconnected lookups:

- Pre SQL
- Post SQL

Guidelines for mappings

Consider the following guidelines when you configure mappings:

- For a target created at runtime, ensure that the Snowflake source does not contain records with the Time data type.
- When you configure filters, consider the following guidelines:
  - If a mapping contains a Filter transformation and also a filter in the Source transformation, the mapping consolidates the filter conditions from both these transformations to filter the records. However, it is recommended that you use only one of these filters at a time in a mapping.
  - You cannot use system variables in filters.
  - You cannot apply a filter for query and multiple source objects.
  - When you configure an IS_date function in an Expression transformation, specify the format for this function. Else, the mapping populates incorrect data.
  - When you configure two Sequence Generator transformations to write to two Snowflake targets, and the sequence objects have the same sequence name in the custom properties, data populates incorrectly.
- For mappings that read from and write to Snowflake, consider the following guidelines:
  - You cannot use a query to read from stored procedures.

- Even if you decrease the precision of the Snowflake String data type in a Source transformation to write to a Snowflake table, the mapping passes without truncating the data.

- When you configure a mapping for source or partial SQL ELT optimization, do not connect the Source transformation to more than one transformation in the mapping. However, in a mapping enabled with full SQL ELT optimization, the Source transformation can branch out to multiple transformations in the mapping pipeline.

- You can configure a custom query in a Source transformation to read from Java or SQL user-defined functions (UDF) in Snowflake.

- When the mapping runs with full or source SQL ELT optimization, some of the queries in the session log are not aliased correctly. The alias for simple queries reflects properly.

- A mapping or mapping in advanced mode fails to read data from multiple tables joined using related objects, where the tables and column names have case-sensitive, special, and unicode characters.

- A mapping that reads from multiple Snowflake objects that do not belong to the same database and schema fails.

- When you use the is_number function, the data populated for some values such as inf, inf and NaN in Snowflake differs with and without SQL ELT optimization applied.

- When you use the IS_NUMBER function in a transformation and the input data contains d or D, for example, in formats such as +3.45d+32 or +3.45D-32, the function returns False or 0.

- When you use the IS_DATE function in a transformation, do not use the J, MM/DD/YYYY SSSSS, MM/DD/Y, and MM/DD/RR formats.

- Mappings that read from or write to Snowflake with multibyte characters in the table or column names might fail. Before you configure a mapping to read from or write data with multibyte characters, set the -DdisablePDOAdvancedAliasing property in the JVM options in the Secure Agent properties.

- When you pass columns with Null values in a Normalizer transformation, Null values are not written to the target.

- When you push the DATE_DIFF() function with the date1 and date2 arguments from a mapping task enabled with the **Create Temporary View** property, the function returns the following different values as compared to a mapping that runs without SQL ELT optimization:

  - The function returns a negative number when the value of date1 is later than the value of date2.

  - The function returns a positive number when the value of date1 is earlier than the value of date2.

  To get the correct return values, set the JVM option to *-DFixSnowflakeDateDiffForPDO=true* for the Secure Agent in Administrator.

- A mapping or mapping in advanced mode fails to write data to the target when the precision of incoming fields exceeds the precision of target fields.

- A mapping enabled for SQL ELT optimization fails when you parameterize the advanced source filter and enclose the parameter value within quotes in the parameter file.
  A mapping configured without SQL ELT optimization with similar filter configurations does not filter data but runs successfully.

Amazon S3 V2 source

The mapping supports the following properties for an Amazon S3 V2 connection:

- Access Key
- Secret Key

The mapping supports the following properties for an Amazon S3 V2 source:

- Source connection parameter
- Source Type - Single, parameter
- Format - Delimited, Avro, ORC, Parquet, and JSON
- Source Type - File and directory
- Folder Path
- File Name
- Compression Format. - Gzip

A mapping enabled for SQL ELT optimization that reads from an Amazon S3 V2 source and writes to a Snowflake target has some restrictions.

Authentication

When you read multiple Avro files using an Amazon S3 connection enabled for IAM authentication, specify the right access key and the secret key in the Amazon S3 connection. For more information, see the help for Amazon S3 V2 Connector.

Create a new target at runtime

A mapping that creates a new target at runtime has the following restrictions:

- To write data from file data types such as Avro, ORC, or Parquet from Amazon S3 to Snowflake, you must delete the **Filename** field.
- Mappings fails with a casting error when the table name contains Unicode characters.

Data types

A mapping has the following restrictions for certain data types:

- You cannot write Avro files that contain special characters.
- You cannot write data that contains the Binary data type.
- You cannot read data in JSON format that contains special characters. For more information about using identifiers, see Identifiers Syntax in the Snowflake documentation.
- If you specify any escape character for the S3 file format, the escape character defaults to backslash.
- ORC files with year 1523 is loaded incorrectly as 1524.
- When you write data with the Time data types from a Parquet file from Amazon S3 to Snowflake, the value of the time differs in the target.
- The precision of JSON data must not exceed the precision of the Snowflake target table.
- If the Amazon S3 source type is a directory and you enable wildcard characters for the directory, the mapping fails. A warning message appears stating that wildcard characters read are not supported with SQL ELT optimization.

For information on how to configure the supported properties, see the Amazon S3 V2 Connector documentation.

Google Cloud Storage V2 source

The mapping supports the following properties for a Google Cloud Storage V2 source connection:

- Source connection, connection parameter
- Source Type - Single, parameter

- Format - Delimited, Avro, Parquet, and JSON
- Google Cloud Storage Path
- Source File Name
- Is Directory

Note: In advanced mode, Google Cloud Storage V2 as the source is not applicable with SQL ELT optimization.

For information on how to configure the supported properties, see the Google Cloud Storage V2 Connector documentation.

Microsoft Azure Data Lake Storage Gen2 source

The mapping supports the following properties for an Microsoft Azure Data Lake Storage Gen2 source connection:

- Account Name
- File System Name

The mapping supports the following properties for a Microsoft Azure Data Lake Storage Gen2 source:

- Source connection, connection parameter
- Source Type - Single, parameter
- Format - Delimited, Avro, Parquet, JSON, and ORC
- Formatting Options
- Filesystem Name Override
- Source Type - File, Directory
- Directory Override - Absolute path; Relative path
- File Name Override - Source object
- Compression Format - GZip to read flat files
- Tracing Level

For information on how to configure the supported properties, see the Microsoft Azure Data Lake Storage Gen2 Connector documentation.

Creating temporary view for source overrides

To push a custom query or an SQL override to Snowflake, the task must create a temporary view based on the provided custom query and then generates the SQL ELT query with the temporary view.

Before you configure a custom query as the source object or you configure an SQL override, perform the following task:

In the **SQL ELT Optimization** section on the **Runtime Options** tab, select **Create Temporary View**.

Note: If you do not set the **Create Temporary View** property, the mapping runs without SQL ELT optimization.

Configuring target copy command options

You can specify the copy command options to load data from Amazon S3 to Snowflake.

Specify the options in the **Additional Write Runtime Parameters** field in the Snowflake Data Cloud advanced target properties of the Target transformation.

When you specify multiple copy command options, separate each option with an ampersand *&*.

Note: The copy option *MATCH_BY_COLUMN_NAME* is not applicable.

For more information about the supported copy command options, see the Snowflake documentation at the following website: https://docs.snowflake.com/en/sql-reference/sql/copy-into-table.html

Configuring SQL ELT optimization

To optimize a mapping, add the mapping to a task, and then configure SQL ELT optimization in the mapping task.

1. Create a mapping task.
2. In the **SQL ELT Optimization** section on the **Runtime Options** tab, set the SQL ELT optimization value to **Full** or **To Source**.
3. If full SQL ELT optimization is not available, select how Data Integration handles SQL ELT optimization in the **SQL ELT Optimization Fallback Option** menu:
   - Partial SQL ELT. Default. Data Integration pushes as much transformation logic as possible to the source and target database. The task processes any transformation logic that it can't push to a database. You can use Partial SQL ELT only when you read from and write to Snowflake.
   - Non SQL ELT. The task runs without SQL ELT optimization.
   - Fail Task. Data Integration fails the task.

   Note: The fallback options are not applicable to mappings in advanced mode.

When you run the mapping task, the transformation logic is pushed to the configured database. To verify that the mapping was optimized, you can check the session log for the job. In Monitor, view the log for jobs.

Context based optimization for multiple targets

When you configure a mapping enabled for full SQL ELT optimization to write to multiple Snowflake targets using a Snowflake Data Cloud connection, you can specify the optimization context for multi-insert and slowly changing dimension type 2 merge scenarios.

You can specify the **Optimization context type** on the **Runtime Options** tab in a task. Based on the optimization context that you specify, Data Integration combines queries issued from multiple targets and constructs a single query for SQL ELT optimization and the task is optimized.

You can enable the following optimization modes in the task properties based on the target operations you specify in the mapping:

Multi-insert

Enable this mode when you insert data from a Snowflake source to multiple Snowflake targets. Data Integration combines the queries generated for each of the targets and issues a single query.

SCD Type 2 merge

Enable this mode when you write to two Snowflake targets, where you use one target to insert data and the other target to update data. Data Integration combines the queries for both the targets and issues a Merge query.

Default is None.

Understanding an SCD type 2 merge mapping

The SCD Type 2 merge mapping uses a Snowflake source and two target transformations that write to the same Snowflake table. One target transformation updates the table while the other transformation inserts data to the Snowflake table

The following image shows a mapping that writes slowly changing dimension data to a Snowflake target table:
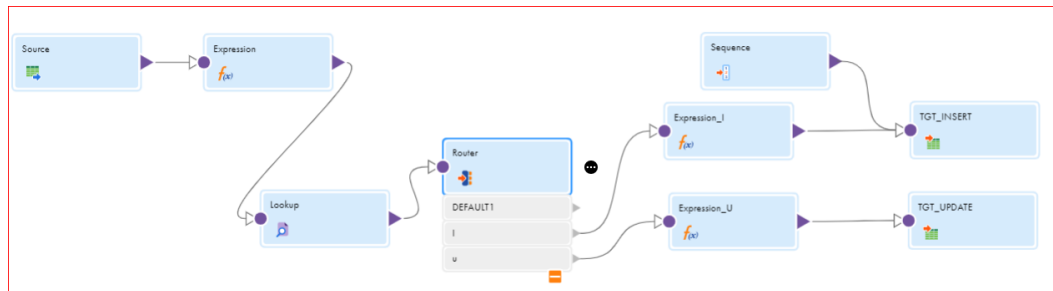
Add lookup and expression transformations to compare source data against the existing target data. You enter the lookup conditions and source columns that you want the Data Integration to compare against the existing target.

For each source row without a matching primary key in the target, the Expression transformation marks the new row. For each source row with a matching primary key in the target, the Expression compares user-defined source and target columns. If those columns do not match, the Expression marks the row changed. The mapping then splits into two data flows.

The first data flow uses the Router transformation to pass only new rows to the Expression transformation. The Expression transformation inserts new rows to the target. A Sequence Generator creates a primary key for each row. The Expression transformation increases the increment between keys by 1,000 and creates a version number of 0 for each new row.

In the second data flow, the Router transformation passes only changed rows to pass to the Expression transformation. The Expression transformation inserts changed rows to the target. The Expression transformation increments both the key and the version number by one.

Restrictions

You cannot use a filter, joiner, and a custom SQL query in an SCD type 2 merge mapping.

Pushing logic to a database with different schemas

You can use cross-schema SQL ELT optimization for a mapping task to read from or write data to Snowflake objects associated with different schemas within the same Snowflake database.

To use cross-schema SQL ELT optimization, create two connections and specify the schema in each connection. Ensure that the schema in the source connection is different from the schema in the target connection, but both the schemas must belong to the same database.

Configuring cross-schema SQL ELT optimization

To push processing logic for different schemas within a Snowflake database, configure cross-schema SQL ELT optimization in the mapping task.

1. Create a mapping task.
    a. Select the configured mapping.
    b. In the **SQL ELT Optimization** section on the **Runtime Options** tab, set the SQL ELT optimization value to **Full**.
2. To enable cross-schema SQL ELT optimization, select **Enable cross-schema SQL ELT optimization**.
    Default is enabled.
3. Save the task and click **Finish**.

Pushing logic to different databases

You can enable cross-database SQL ELT optimization to run queries on data spread across multiple Snowflake databases.

You can configure cross-database SQL ELT optimization in the mapping task. Ensure that the Snowflake source and target transformations in the mapping uses two different Snowflake Data Cloud connections, or Snowflake ODBC connections.

Configuring cross-database SQL ELT optimization

To push processing logic across different Snowflake databases, configure cross-database SQL ELT optimization in the mapping task.

1. Create a mapping task.
    a. Select the configured mapping.
    b. In the **SQL ELT Optimization** section on the **Runtime Options** tab, set the SQL ELT optimization value to **Full**.
2. To enable push down across databases, in the **Advanced Options** on the **Runtime Options** tab, select **Allow SQL ELT across Databases** as the **Session Property Value** and set the **Session Property Value** to **Yes**.

Clean stop a SQL ELT optimization job

When a task enabled for SQL ELT optimization is running, you can clean stop the job to terminate all the issued statements and processes spawned by the job.

Use the **Clean Stop** option on the My Jobs page in Data Integration and the All Jobs and Running Jobs page in Monitor.

See the following exceptions before you clean stop a SQL ELT optimization task:

- When you clean stop a task enabled for source SQL ELT optimization that reads from or writes to Snowflake and the target or source properties in the mapping contains pre-SQL or post-SQL statements, the job continues to run the target post-SQL query even though the select query is terminated.

- When you run a mapping configured to create a new target at runtime and clean stop the job immediately, Data Integration creates the target table even if the job is terminated.

Verify the SQL ELT query in the session log

To verify that the SQL ELT optimization was applied while running the mapping, you can check the session log for the job. In Monitor, view the log for jobs.

Check the queries in the session logs to verify if the mapping applied SQL ELT optimization.

For example, the following query is generated in the session log for a mapping enabled with full SQL ELT optimization:

```
39   OPT_63306 [2020-08-06 08:57:23.875] [Full Pushdown optimization is supported between the source and the target system.].
40   OPT_63306 [2020-08-06 08:57:23.875] [Validating mapping for Pushdown Optimization.].
41   OPT_63309 [2020-08-06 08:57:23.875] Pushdown Optimization was successfully enabled.

86   MAPPING> SNOWFLAKECLOUDDATAWAREHOUSE_10000 [2020-08-06 08:57:40.523] [INFO] Inserting data into Snowflake target table : INSERT INTO
     "DB_PC_AUTO"."SCHEMA_PC_AUTO"."TEST1_TGT"("F1_INT") SELECT (SINH((t0."F1_INT"::NUMBER(38,0))::DOUBLE)::NUMBER(18,0))::DOUBLE FROM "CQA"."CQA_SCHEMA"."TEST1_SRC"
     AS t0
```

In the example, the generated SQL includes both the `Insert Into` and `Select` queries pushed down to the database as a single statement.

The session log provides the SQL ELT status. You can check the details to troubleshoot the error.

For example, the session log shows the following error details in the query:

```
OPT_63306 [2020-09-10 14:09:05.716] [Source level filter is not supported for pushdown optimization.].
OPT_63307 [2020-09-10 14:09:05.716] Pushdown optimization failed at Analyze phase.
OPT_63310 [2020-09-10 14:09:05.716] Failed to enable Pushdown Optimization.
```

When you do not enable SQL ELT optimization in a mapping, separate select and insert statements are generated for the read and write operations:

```
READER_1_1_1>SNOWFLAKECLOUDDATAWAREHOUSE_1000 [2020-09-10 14:09:29.4781] [INFO]
The Snowflake Connector uses the following SQL query to read data: SELECT "DEPTID",
"DEPTNAME" FROM "DEPT" WHERE
( 'DEPT'."DEPTID" >=103) ORDER BY "DEPT"."DEPTOD" desc
```

When you enable SQL ELT optimization in a mapping in advanced mode, use the following sample query to read from and write to Snowflake:

```
INSERT INTO "DB_PC_AUTO"."SCHEMA_PC_AUTO"."ALL_DATA_TYPES_TARGET"("ID")
    SELECT
          ALL_0."ID" as c0
    FROM "DB_PC_AUTO"."SCHEMA_PC_AUTO"."ALL_DATA_TYPES"
     AS ALL_0
```

Troubleshooting

When you run a mapping to write data from a string column that contains date and time information, the mapping writes date and time information incorrectly to the Snowflake target.

To resolve this issue, set the JVM option to `-DHonorInfaDateFormat=true` for the Secure Agent.

To configure the JVM option in Administrator, perform the following steps:

1. Select **Administrator** > **Runtime Environments**.
2. On the **Runtime Environments** page, select the Secure Agent machine that runs the mapping.
3. Click **Edit**.
4. In the **System Configuration Details** section, select **Data Integration Server** as the **Service** and **DTM** as the **Type**.
5. Edit the **JVMOption** system property and set the value to `-DHonorInfaDateFormat=true`.
6. Click **Save**.

Data type reference

Cloud Data Integration uses the following data types in Snowflake Data Cloud mappings and mappings in advanced mode with or without SQL ELT optimization enabled.

- Snowflake native data types appear in the source and target transformations when you choose to edit metadata for the fields.

- Transformation data types. Set of data types that appear in the transformations. These are internal data types based on ANSI SQL-92 generic data types, which the agent uses to move data across platforms. They appear in all transformations in a mapping or mapping in advanced mode.

When the agent reads source data, it converts the native data types to the comparable transformation data types before transforming the data. When the agent writes to a target, it converts the transformation data types to the comparable native data types.

Snowflake Data Cloud and transformation data types

The following table lists the Snowflake data types that Data Integration supports and the corresponding transformation data types:

Snowflake Data Cloud Data Type	Transformation Data Type	Range and Description
Binary (Varbinary)	binary	Maximum value: 8,388,60 Default value is 8,388,60.
Boolean	string	A Boolean attribute.
Date	datetime	Date and time values.
Float (Double, Double precision, Real, Float, Float4, Float8)	double	Floating point numbers with double-precision (64 bit). Maximum value: 1.7976931348623158e+307 Minimum value: -1.79769313486231E+307
Number (Decimal, Numeric)	decimal	Number with 28-bit precision and scale for mappings. Number with 38-bit precision and scale for mappings in advanced mode.

Snowflake Data Cloud Data Type	Transformation Data Type	Range and Description
NUMBER (Int, Integer, Bigint, Smallint, Tinyint, Byteint)	decimal	Number with 28-bit precision and scale as 0. Maximum value: 9.99999999999999E+27 Minimum value: -9.99999999999999E+26
Time	datetime	Date and time values. This data type doesn't apply to mappings in SQL ELT mode and mappings in advanced mode.
Timestamp_LTZ	datetime	Date and time values.
Timestamp_NTZ (Timestamp_NTZ, datetime)	datetime	Date and time values.
Timestamp_TZ	datetime	Date and time values.
Varchar (Text, Char, Character, String)	string	Maximum value: 16,777,216 Default value is 16,777,216.

Semi-structured data types and transformation data types

Snowflake uses the Variant data type to store and represent semi-structured data. The Variant data type can store values of any other type, including object and array, up to a maximum size of 16 MB uncompressed.

Array, object, and variant from the Snowflake source are mapped to the String data type in Cloud Data Integration. While writing to the target, these strings can be written as Array, Object, or Variant columns to the Snowflake target. The strings that you write to Snowflake must be in a serialization format, just as they appear after the read operation.

When you use the Create New at Runtime option to write the Variant data type from a source to the Snowflake target, Data Integration writes Variant as Varchar to the target. You can edit the field mapping to map Varchar to Variant before you write to the target. In a completely parameterized mapping, you cannot edit the target metadata from the default Varchar data type to variant.

The following table lists the semi-structured data types that you can read from Snowflake and the corresponding transformation data types that these map to in Cloud Data Integration:

Snowflake Data Type	Transformation Data Type	Description and Range
Array	String	16,777,216
Object	String	16,777,216
Variant	String	16,777,216

Note: The default size to read or write semi-structured data types is set to 65536 bytes. To increase the limit, add the following parameter and set the required value in the **Additional JDBC URL Parameters** field of the Snowflake Data Cloud connection properties: *semiStructuredDTPrecision=<size>*

Rules and guidelines for data types

When you read or write data, some differences in processing and configuration apply for certain data types.

Mappings

Consider the following rules and guidelines for mappings:

- You can read or write data of Binary data type that is in Hexadecimal format.
- The agent reads or writes the maximum float value *1.7976931348623158e+308* as infinity.
- You can use the following formats to specify filter values of the Datetime data type:
  - YYYY-MM-DD HH24:MI:SS
  - YYYY/MM/DD HH24:MI:SS
  - MM/DD/YYYY HH24:MI:SS
- If a Snowflake Cloud Data lookup object contains fields with the String data type of maximum or default precision and the row size exceeds the maximum row size, the task fails.
- The performance of a write operation slows down if the data contains the Date fields.
- A task that captures changed data from a CDC source fails when the Snowflake target contains a repeated column of the Record data type.
- When you handle dynamic schemas in mappings, the following updates are not applicable:
  - Schema updates to the Timestamp and Date data types.
  - Schema updates that involve a decrease in the precision of the Varchar data type.

Mappings in advanced mode

Consider the following rules and guidelines for mappings in advanced mode:

- When you read Time data types, you must map the Time to Timestamp in the SQL override for column of type Time. For example, see the sample SQL override query:
  ```
  SELECT "C1_DATE", to_timestamp (to_char("C2_TIME_3" , 'HH24:MI:SS.FF'), 'HH24:MI:SS.FF')
  AS "C2_TIME_3", to_timestamp (to_char("C3_TIME_5" , 'HH24:MI:SS.FF'), 'HH24:MI:SS.FF') AS
  "C3_TIME_5", to_timestamp (to_char("C4_TIME" , 'HH24:MI:SS.FF'), 'HH24:MI:SS.FF') AS
  "C4_TIME" FROM "SALES"."SF_DEV"."SRC_DATE_TIME"
  ```
- When you handle dynamic schemas, the following updates are not applicable:
  - Schema updates that involve a decrease in the precision of the Varchar data type.
  - Schema updates to the Timestamp and Date data types.
  - When you use the alter and apply changes option to write to the Snowflake target, Data Integration ignores the changes made to the Timestampntz, Datetime, Timestamptz, Timestampltz, and Boolean data types from the source schema.

Additional runtime configurations

You can configure additional attributes for some of the Snowflake Data Cloud tasks in the Secure Agent properties.

You can enable bulk processing of records, set logging for large records, configure a local staging directory, optimize data staging, and improve memory requirements for read operations.

Configure JVM memory requirements

Specify the Java heap space memory as *-Xmx256m* in the Secure Agent properties to avoid the JDBC driver internal errors in mappings and mapping tasks that read data from Snowflake:

This update does not apply for mappings configured with SQL ELT optimization using the Snowflake ODBC connection.

Perform the following steps to configure the JVM memory:

1. In Administrator, select the Secure Agent listed on the **Runtime Environments** tab.
2. Click **Edit**.
3. In the **System Configuration Details** section, select **Data Integration Service** as the service and **DTM** as the type.
4. Edit the **JVMOption1** property, and enter *-Xmx256m*.
5. Click **Save**.

Configure bulk processing

You can enable bulk processing to write large amounts of data to Snowflake. Bulk processing utilizes minimal number of API calls and the performance of the write operation is optimized.

To enable bulk processing, specify the property *-DENABLE_WRITER_BULK_PROCESSING=true* in the Secure Agent properties:

Perform the following steps to configure bulk processing before you run a mapping:

1. In Administrator, select the Secure Agent listed on the **Runtime Environments** tab.
2. Click **Edit**.

3. In the **System Configuration Details** section, select **Data Integration Server** as the service and **DTM** as the type.

4. Edit the JVM option, and enter -*DENABLE_WRITER_BULK_PROCESSING=true*.

5. Click **Save**.

Note: This update does not apply for mapping tasks configured with SQL ELT optimization using the Snowflake ODBC connection or the Snowflake Data Cloud connection.

Configure logging properties

Error messages might appear in the session log from Snowflake in some scenarios.

To avoid this error, perform the following tasks:

1. Navigate to the following location:
    ```
    <Secure Agent installation directory>\apps\jdk\1.8.0_Zulu<version>\jre\lib
    ```

2. Set the `logging.properties` field to include `java.util.logging.ConsoleHandler.level = WARNING`

If you do not set this logging property, the following errors might appear in the session logs:

`net.snowflake.client.jdbc.internal.apache.http.impl.execchain.RetryExec execute`

`INFO: I/O exception (net.snowflake.client.jdbc.internal.apache.http.NoHttpResponseException) caught when processing request to {s}...> The target server failed to respond.`

Configure the temp directory for a mapping

The Secure Agent creates the local staging files in a default temp directory. You can configure a different directory to store the local staging files.

To configure a different directory for the local staging files, perform the following steps:

1. In Administrator, click **Runtime Environments**.

    The Runtime Environments page appears.

2. Select the Secure Agent for which you want to set the custom configuration property.

3. Click **Edit Secure Agent** icon corresponding to the Secure Agent you want to edit in **Actions**.

    The Edit Secure Agent page appears.

4. Select the **Service** as **Data Integration Server** in the **System Configuration Details** section.

5. Select the **Type** as **DTM** in the **System Configuration Details** section.

6. Set the JVM option to `-Djava.io.tmpdir=E:\Snowflake\temp`.

7. Click **Save**.

8. Restart the Secure Agent.

Optimize the staging performance of a mapping

Data Integration, by default, creates a flat file locally in a temporary folder to stage the data before reading from or writing to Snowflake. You can set Data Integration to optimize the staging performance of the read and write operations.

If you do not set the staging property, Data Integration performs staging without the optimized settings, which might impact the performance of the task.

Setting the staging property

You can optimize the mapping performance of both the read and write operations.

You need to set the following staging property in the agent properties based on the operation that you want to optimize:

- To optimize the read operation, set the following property: `INFA_DTM_RDR_STAGING_ENABLED_CONNECTORS`
- To optimize the write operation, set the following property: `INFA_DTM_STAGING_ENABLED_CONNECTORS`

Perform the following tasks to set the staging property for the Tomcat in the Secure Agent properties:

1. In Administrator, click **Runtime Environments**.
2. Edit the Secure Agent for which you want to set the property.
3. In the **System Configuration Details** section, select the **Service** as **Data Integration Server** and the type as **Tomcat**.
4. Set the value of the Tomcat property to the plugin ID of Snowflake Data Cloud Connector.
   You can find the plugin ID in the manifest file located in the following directory: `<Secure Agent installation directory>/downloads/<Snowflake package>/CCIManifest`

   The following image shows the property set for the Secure Agent for the read and write operations:

   | Tomcat | INFA_DTM_STAGING_ENABLED_CONNECTORS | 305501 |
   | Tomcat | INFA_DTM_RDR_STAGING_ENABLED_CONNECTORS | 305501 |

When you run the mapping, the flat file is created in the following directory in your machine based on the operation for which you set the property:

- Read operation: `C:\Windows\Temp\StagingReader\<Source_Name><data_.csv>`
- Write operation: `C:\Windows\Temp\snowflake\stage\<Snowflake_Target.txt>`

You can check the session logs.

- Read operation: If the staging is done through the flat file successfully, Data Integration logs the following message in the session log: `Staging mode is enabled to read data`.
- Write operation: If the staging is done through the flat file successfully, Data Integration logs the following message in the session log: `The INFA_DTM_STAGING is successfully enabled to use the flat file to create local staging files.`

Rules and guidelines

When you enable the staging property for the read operation, you cannot configure a custom query to call a stored procedure from Snowflake. If you run a query to call a stored procedure, the task fails.

Consider the following guidelines when you enable the staging property for the write operation:

- If you run a mapping enabled for SQL ELT optimization, the mapping runs without SQL ELT optimization.
- If the data contains timestamp data types with time zone, the job runs without staging the data in the local flat file.

- If the mapping contains Oracle CDC as the source and Snowflake as the target, the job runs without staging the data in the local flat file.
- You can determine the behavior of writing data with empty and null values to the target when you enable the staging property and set the copyEmptyFieldAsEmpty in the **Additional Write Runtime Parameters** field in the Target transformation properties.
  The table describes the behavior of empty and null values when you set these properties:

Additional Write Runtime Parameters	Enable staging optimization	Disable staging optimization
copyEmptyFieldAsEmpty=FALSE	Empty -> NULL NULL -> NULL	Empty -> NULL NULL -> NULL
copyEmptyFieldAsEmpty=TRUE	Empty -> Empty NULL -> Empty	Empty -> Empty NULL -> Empty
Property not set	Empty -> Empty NULL -> NULL	Empty -> Empty NULL -> NULL

For information on the performance comparison of the execution time before and after setting the staging property, see the following How-To Library article:
Performance Tuning and Best Practices for Snowflake Data Cloud Connector

Optimize the write operation in a CDC mapping

You can optimize the write operation in a mapping that reads from a CDC source and writes to a Snowflake target.

To optimize the write operation, set the JVM option to *-DSNOWFLAKE_CDC_PARALLEL_EXECUTION=true* for the Secure Agent.

1. Log in to Informatica Intelligent Cloud Services.
2. Select **Administrator** > **Runtime Environments**.
3. On the **Runtime Environments** page, select the Secure Agent machine that runs the mapping.
4. Click **Edit**.
5. In the **System Configuration Details** section, select **Data Integration Server** as the service and **DTM** as the type.
6. Edit the JVM option, and set the value to *-DSNOWFLAKE_CDC_PARALLEL_EXECUTION=true*.

7. Click **Save**.

Upgrading to Snowflake Data Cloud Connector

If you are currently using the Snowflake V1 connection or the Snowflake ODBC connection to access and perform operations in Snowflake, you can upgrade to Snowflake Data Cloud Connector.

As the object type is compatible, you can switch the source or target connection type in existing mappings that use the Snowflake connection or the Snowflake ODBC connection with the Snowflake Data Cloud connection.

After you replace the connection in the mapping, the object selected previously is not retained. You must reimport the Snowflake object. The configured advanced source, target, and lookup properties in the fields that are common between the two connectors are retained in the new connector. You also have the option to retain the configured field mappings from the old connector.

You can run the mapping successfully using the configured values from the old connector. You can additionally configure features that the enhanced Snowflake Data Cloud Connector offers.

Important: You can also migrate a Snowflake V1 connection to Snowflake Data Cloud connection in existing assets using the Data Integration REST API. For more information, see the topic "connectionMigration" in the *REST API Reference*. For details on the migration process and rules and guidelines, see the following How-To Library article: [Migrating a connector from previous versions using the Data Integration REST API](#)

Connection switching example

You want to switch your existing Snowflake mapping that uses the Snowflake V1 connection to the Snowflake Data Cloud connection.

1. Open the existing Snowflake V1 mapping that you want to switch to Snowflake Data Cloud.

   The following image shows an existing mapping that uses the Snowflake V1 connection and contains the configured advanced properties in the Target transformation:

   The configured target object path in this example is: `CQA/CQA_SCHEMA/AM_CUSTOMER_SRC`

2. To retain the mapped fields from the field mapping when you switch the connection, on the **Field Mapping** tab, choose from the following **Field Map Options** menu in the Snowflake V1 mapping:

   - To retain the fields automatically mapped after the switch, select **Automatic**.

   - To manually map the retained fields after the switch, select **Manual**.

     Note: When you select manual, after switching the connection, you have the option to automap the retained fields using the previous mapping.

3. To switch the connection, in the **Connection** field, change the connection from Snowflake V1 to Snowflake Data Cloud.

4. In the **Change Connection** dialog box, select the following properties, and click **Yes**:
   - **Connection switch**. Switches to the connection that you select.
   - **Retain field mapping**. Retains the configured field mappings from Snowflake V1.

   The following image shows the options that you must select:
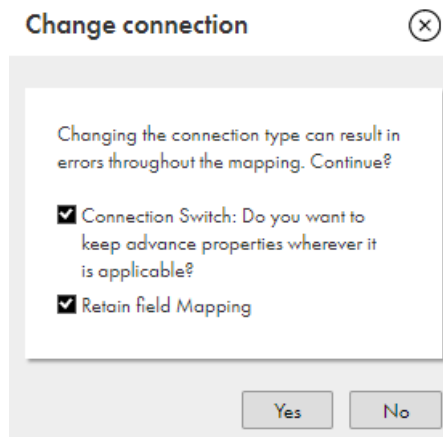
5. Use the same object path in the mapping as Snowflake V1.

   The following image shows the switched connection with the same object path: `CQA/CQA_SCHEMA/AM_CUSTOMER_SRC`
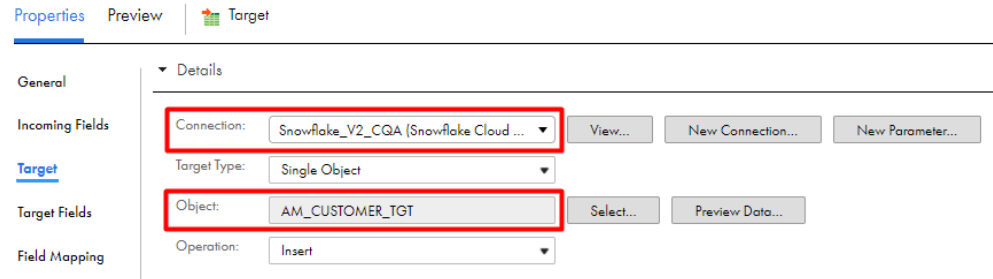
   The configured target advanced properties from the Snowflake V1 mapping reflect in the Target transformation.

6. If you had selected **Manual** on the **Field Mapping** tab in the Snowflake V1 mapping and you want to reflect the field mappings in Snowflake Data Cloud, on the **Field Mapping** tab, select **Automap**, and then select **Previous Mapping**.
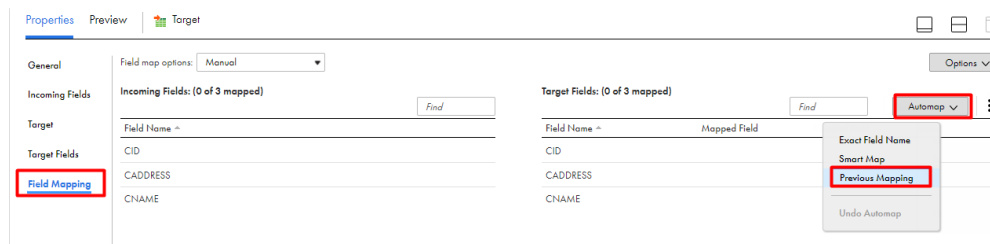
   Note: If you had selected **Automatic** in Snowflake V1, you do not have to perform this task.

In the following image, the configured mapped fields from the Snowflake V1 mapping reflects in the Snowflake Data Cloud mapping:
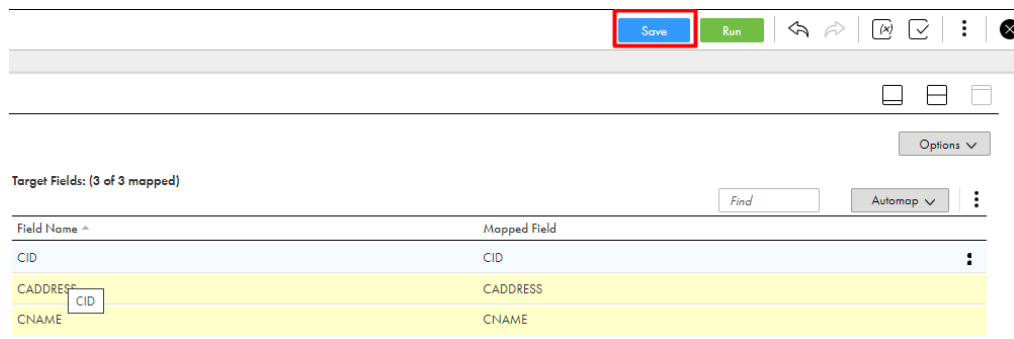
7.   Click **Save**.

Properties retained after the switch

When you replace the source or target connection type in existing mappings or mapping tasks that use the Snowflake V1 connection or the Snowflake ODBC connection with the Snowflake Data Cloud connection, you have the option to retain the configured advanced properties and the field mappings.

The following table lists the configured advanced source, lookup, and target properties from the Snowflake ODBC connection that are retained in Snowflake Data Cloud mappings:

Properties	Snowflake ODBC properties retained in Snowflake Data Cloud
Source	- Pre SQL - Post SQL
Target	- Pre SQL - Post SQL - Truncate target - Forward rejected rows - Update override

The following table lists the configured advanced source, lookup, and target properties from Snowflake V1 retained in Snowflake Data Cloud mappings:

Properties	Snowflake V1 properties retained in Snowflake Data Cloud
Source	- Database - Schema - warehouse - Role - Pre SQL - Post SQL - Table name - SQL override - Tracing level
Lookup	- Database - Schema - Warehouse - Role
Target	- Database - Schema - Warehouse - Role - Pre SQL - Post SQL - Batch row size - Number of local staging files - Truncate target table - Additional runtime properties - Table name - Rejected file path - Success file directory - Error file directory - Forward rejected rows

Rules and guidelines

Consider the following rules before you replace the Snowflake ODBC connection or Snowflake V1 connection in an existing mapping with the Snowflake Data Cloud connection:

- The filter and sort values are not retained. You must manually add the sort and filter values after you switch to the new connection.

- If the existing mapping contains multiple objects, the relationship between these objects is not preserved after you switch to the new connection.

Index