Informatica® PowerExchange for Snowflake 10.4.0

User Guide

Informatica PowerExchange for Snowflake User Guide
10.4.0
December 2019

Publication Date: 2021-04-04

Table of Contents

Preface

Use the *Informatica® PowerExchange® for Snowflake User Guide* to learn how to read from and write to Snowflake by using the Developer tool. Learn to create a Snowflake connection, develop and run mappings in the native, Hadoop, or Databricks environment.

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 Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

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 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 Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

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

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Introduction to PowerExchange for Snowflake

This chapter includes the following topics:

PowerExchange for Snowflake Overview

You can use PowerExchange for Snowflake to extract data from and load data to Snowflake. You can also read data from and write data to Snowflake that is enabled for staging data in Azure or Amazon.

You can use Snowflake objects as sources and targets in mappings. When you use Snowflake objects in mappings, you must configure properties specific to Snowflake. You can also use Snowflake sources and targets in dynamic mappings.

You can validate and run Snowflake mappings in the native or in the non-native environment such as Hadoop and Databricks. In the Hadoop environment, you can run mappings on the Spark engine. You can also run profiles against Snowflake objects in the native environment. PowerExchange for Snowflake uses the Snowflake Spark Connector APIs to run Snowflake mappings on the Spark engine. To run mappings in the native environment, PowerExchange for Snowflake uses the Snowflake loader APIs.

You can also use Snowflake as a target in a streaming mapping. For more information, see the *Data Engineering Streaming User Guide*.

Example

An enterprise application uses an Oracle database to store the product transaction details such as transactionID, customerID, productID, quantity, and order date. You need to analyze the completed transactions, pending transactions, and availability of stock. Use PowerExchange for Snowflake to create a mapping to extract all the transaction records from the Oracle source, and load the records to a Snowflake target for data analysis.

Introduction to Snowflake

Snowflake is a cloud data warehouse service that organizations can use to store and analyze data.

Snowflake is a Software-as-a-Service (SaaS) application that uses an SQL database engine with an architecture designed for the cloud.

The Snowflake architecture comprises the following three layers, which sets Snowflake apart from other data warehouses with its additional functionalities and capabilities:

- Database Storage. When you load data to Snowflake, Snowflake reorganizes and stores the data in the Snowflake database. You can access the data stored in the Snowflake database through SQL query operations.
- Query Processing. Snowflake processes all queries in the query processing layer. The processing layer contains all the compute resources that Snowflake needs to run queries. For example, Snowflake uses CPU, memory, and temporary storage to run queries.
- Cloud Services. The cloud services layer contains all the different components that Snowflake needs to process the user requests. For example, Snowflake uses authentication, infrastructure management, and access control services to process user requests.

PowerExchange for Snowflake Configuration

PowerExchange for Snowflake installs with Informatica. You enable PowerExchange for Snowflake with a license key.

Prerequisites

Before you use PowerExchange for Snowflake, complete the following prerequisite tasks:

- Install or configure Informatica Services. Verify that the domain has a Data Integration Service and a Model Repository Service.
- Verify that you have write permissions on all the directories within the `<INFA_HOME>` directory.
- Get a license for PowerExchange for Snowflake.
- If you want to run Snowflake mappings on a Kerberos-enabled Hadoop cluster, ensure that you have permissions to create a directory in the cluster.
  If you do not have permissions and you run a Snowflake mapping on a Kerberos-enabled cluster, the Data Integration Service fails to create the cache folder on the cluster and the mapping fails.

For more information about product requirements and supported platforms, see the Product Availability Matrix on Informatica Network:
https://network.informatica.com/community/informatica-network/product-availability-matrices

CHAPTER 2

Snowflake Connections

This chapter includes the following topics:

Snowflake Connections Overview

Use a Snowflake connection to access a Snowflake database.

Use the Snowflake connection to import Snowflake metadata, create data objects, preview data, and run mappings. When you create a Snowflake connection, you define the connection attributes that the Developer tool uses to connect to the Snowflake database.

Use the Developer tool, Administrator tool, or infacmd to create a Snowflake connection.

Snowflake Connection Properties

When you set up a Snowflake connection, you must configure the connection properties.

Note: The order of the connection properties might vary depending on the tool where you view them.

The following table describes the Snowflake connection properties:

Property	Description
Name	The name of the connection. The name is not case sensitive and must be unique within the domain. You can change this property after you create the connection. The name cannot exceed 128 characters, contain spaces, or contain the following special characters:~ ` ! $ % ^ & * ( ) - + = { [}] | \ : ; " ' < , > . ? /
ID	String that the Data Integration Service uses to identify the connection. The ID is not case sensitive. The ID must be 255 characters or less and must be unique in the domain. You cannot change this property after you create the connection. Default value is the connection name.

Property	Description
Description	Optional. The description of the connection. The description cannot exceed 4,000 characters.
Location	The domain where you want to create the connection.
Type	The connection type. Select SnowFlake.
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.
Warehouse	The Snowflake warehouse name.
Role	The Snowflake role assigned to the user.

Additional JDBC URL Parameters	Enter one or more JDBC connection parameters in the following format:
	`<param1>=<value>&<param2>=<value>&<param3>=<value>....`
	For example:
	`user=jon&warehouse=mywh&db=mydb&schema=public`
	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 supported.
	For more information about configuring Okta authentication, see the following website:
	https://docs.snowflake.net/manuals/user-guide/admin-security-fed-auth-configure-snowflake.html#configuring-snowflake-to-use-federated-authentication

Creating a Snowflake Connection

Create a Snowflake connection before you create a Snowflake data object.

1. In the Developer tool, click **Window** > **Preferences**.
2. Select **Informatica** > **Connections**.
3. Expand the domain in the **Available Connections**.
4. Select the connection type **Database** > **SnowFlake**, and click **Add**.
5. Enter a connection name and an optional description.
6. Select SnowflakeConnection as the connection type.
7. Click **Next**.
8. Configure the connection properties.
9. Click **Test Connection** to verify the connection to Snowflake.
10. Click **Finish**.

CHAPTER 3

PowerExchange for Snowflake Data Objects

This chapter includes the following topics:

Snowflake Data Object Overview

A Snowflake data object is a physical data object that uses Snowflake as a source or target. A Snowflake data object is a physical data object that represents data based on a Snowflake resource.

You can configure the data object read and write operation properties that determine how you can read data from Snowflake sources and load data to Snowflake targets.

Create a Snowflake data object from the Developer tool. PowerExchange for Snowflake creates the data object read operation and data object write operation for the Snowflake data object. You can edit the advanced properties of the data object read or write operation and run a mapping.

Snowflake Data Object Properties

Specify the data object properties when you create the data object.

The following table describes the properties that you configure for the Snowflake data objects:

Property	Description
Name	Name of the Snowflake data object.
Location	The project or folder in the Model Repository Service where you want to store the Snowflake data object.
Connection	Name of the Snowflake connection.

Snowflake Data Object Read Operation

Create a mapping with a Snowflake data object read operation to read data from Snowflake.

Snowflake Data Object Read Operation Properties

Snowflake data object read operation properties include run-time properties that apply to the Snowflake data object.

The Developer tool displays advanced properties for the Snowflake data object operation in the Advanced view.
The following table describes the advanced properties for a Snowflake data object read operation:

Property	Description
Database	Overrides the database name specified in the connection.
Schema	Overrides the schema name specified in the connection.
Warehouse	Overrides the Snowflake warehouse name specified in the connection.
Role	Overrides the Snowflake user role specified in the connection.
Pre SQL	SQL statement that the Data Integration Service executes before extracting data from the source. For example, if you want to update records in the database before you extract the records from the table, specify a pre-SQL statement.
Post SQL	SQL statement that the Data Integration Service executes after extracting data from the source. For example, if you want to delete some records after the latest records load, specify a post-SQL statement.
Table Name	Overrides the table name of the imported Snowflake source table.
SQL Override	Overrides the default SQL query used to extract data from the Snowflake source.

Query Properties

Use the **Query** tab to specify the join, filter, and sort conditions.

You can configure the following query properties for Snowflake sources:

Select Distinct

Selects unique values from the source. The Data Integration Service filters out unnecessary data when you use the Snowflake data object in a mapping.

Join

The Join expression you specify in a read operation to join data from multiple sources in a Snowflake data object.
Select the sources on the **Data Object** tab and define the join condition.

You can use the native or platform expression to join specific records:

- When you configure a platform expression, in the **Tables** section, you can select the outer, inner, left, and right joins to join the tables.
- In the Relationship section, define the relationship for the join:
  - Left Field. The column on which you want to apply the join condition.
  - Operator. The operators that you can use to join tables. You can select one of the following operators: =, >, >=, <, <=, !=
  - Right Field. The value you specify to join the tables.
- When you configure a native expression, select the fields and define a join query syntax in the Join text field. 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 native expression is appended to the join condition.

Filter

The filter value in a read operation. The filter specifies the WHERE clause of the SELECT statement. Use a filter to reduce the number of rows that the Data Integration Service reads from the Snowflake source. When you enter a source filter, the Integration Service adds a WHERE clause to the default query.
You can use the native or platform expression to filter specific records.

- Configure the platform expression, you can specify the filter using the following fields:
  - Left Field. The column on which you want to apply the filter condition.
  - Operator. The operators that you can use to filter tables. You can select one of the following operators: =, >, >=, <, <=, !=
  - Right Field. The value based on which you want to filter the records.
- When you configure a native expression, select the fields and define a filter query syntax in the Filter text field. The condition you specify in the text box for the native expression is appended to the filter condition.

Sort

Sorts the rows queried from the Snowflake source. The Data Integration Service adds the ports to the ORDER BY clause in the default query.

Snowflake Data Object Write Operation

Create a mapping to write data to Snowflake. Use the Snowflake connection, and define the write operation properties to write data to Snowflake.

You can perform insert, update, delete, and upsert operations on a Snowflake target.

Snowflake Data Object Write Operation Properties

Snowflake data object write operation properties include run-time properties that apply to the Snowflake data object.

The Developer tool displays advanced properties for the Snowflake data object operation in the Advanced view.

The following table describes the Advanced properties for a Snowflake data object write operation:

Property	Description
UpdateMode	Loads data to the target based on the mode you specify. Select one of the following modes: - Update As Update. Updates all rows flagged for update. - Update Else Insert. Updates all rows flagged for update if they exist in the target and then inserts any remaining rows marked for insert.
Database	Overrides the database name specified in the connection.
Schema	Overrides the schema name specified in the connection.
Warehouse	Overrides the Snowflake warehouse name specified in the connection.
Role	Overrides the Snowflake user role specified in the connection.
Pre SQL	SQL statement that the Data Integration Service executes before extracting data from the source. For example, if you want to assign sequence object to a primary key field of the target table before you load data to the table, specify a pre-SQL.
Post SQL	SQL statement that the Data Integration Service executes after extracting data from the source. For example, if you want to alter the table created by using create target option and assign constraints to the table before you load data to the table, specify a post-SQL.
Batch Row Size	The number of rows that the Data Integration Service writes to a file. When the number of rows written to the file reaches the value specified, the Data Integration Service flushes the data queue and starts processing the write commands
Number of local staging files	The number of files that represents a single batch of data. The default number of files is 64. After the Data Integration Service uploads the specified number of local staging files to the Snowflake user stage, Snowflake unloads the data to the target table.
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	Specify additional runtime parameters.
	For example, if you want to specify the user-defined stage in the Snowflake database to upload the local staging files, specify the name of the stage location in the following format:
	`remoteStage=REMOTE_STAGE`
	If you want 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, for example:
	`Compression=On`
	By default, compression is on.
	Separate multiple runtime parameters with &.

Property	Description
Table Name	Overrides the table name of the Snowflake target table.
Rejected File Path	The filename and path of the file where the Data Integration Service writes rejected records. For example, `\rejectedfiles\reject7` Applicable only in the native environment.
Target Schema Strategy	Target schema strategy for the Snowflake target table. You can select one of the following target schema strategies: - RETAIN - Retain an existing target schema. - CREATE - Create a target if it does not exist. - **Note:** To use the CREATE option, you must provide the value of the database and schema name property in the Snowflake connection. - Assign Parameter.

Configuring 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 date size is 50 MB, calculate the csvFileSize value in bytes, for example 50*1024*1024 and then specify 52428800 as the csvFileSize. It is recommended that you configure the right combination of the number of local staging files and csvFileSize while writing data to Snowflake for better performance.

Configure Loader Properties as Additional Runtime Parameters

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

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

Property	Supported Type	Description	Default Value
oneBatch	Boolean	Process all data in a single batch.	false
remoteStage	String	Specifies to use internal stage. **Note:** External stage is not supported.	"~"(user stage)
onError	String	Specifies the action to perform when an error is encountered while loading data from a file. For example, `on_error option ABORT_STATEMENT|CONTINUE|SKIP_FILE`	CONTINUE
compressFileByPut	Boolean	Compress file by PUT.	false

Property	Supported Type	Description	Default Value
compressDataBeforePut	Boolean	Compress data before PUT. The loader compresses the data to a gzip format before uploading the data.	true
copyEmptyFieldAsEmpty	Boolean	The COPY command option to set incoming empty fields as null.	-

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

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

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.

Snowflake Data Object Lookup Operation

Create a mapping with a Snowflake data object lookup operation to lookup data from Snowflake based on a lookup condition.

Snowflake Data Object Lookup Operation Properties

Snowflake data object lookup operation properties include run-time properties that apply to the Snowflake data object.

The Developer tool displays run-time properties for the Snowflake data object lookup operation in the Run-time view.

The following table describes the run-time properties for a Snowflake data object lookup operation:

Property	Description
Lookup caching enabled	Indicates whether the Data Integration Service caches lookup values. Select **Lookup caching enabled** to enable lookup caching. When you enable lookup caching, the Integration Service queries the lookup source once, caches the values, and looks up values in the cache. Caching the lookup values can increase performance on large lookup tables. When you disable caching, each time a row passes into the transformation, the Integration Service issues a select statement to the lookup source for lookup values. **Note:** The Integration Service supports lookup caching on the native and non-native environments. The Integration Service does not support uncached lookup in the non-native environment.

The Developer tool displays advanced properties for the Snowflake data object lookup operation in the Advanced view.
The following table describes the advanced properties for a Snowflake data object read operation:

Property	Description
Database	Overrides the database name specified in the connection.
Schema	Overrides the schema name specified in the connection.
Warehouse	Overrides the Snowflake warehouse name specified in the connection.
Role	Overrides the Snowflake user role specified in the connection.
Pre SQL	SQL statement that the Data Integration Service executes before extracting data from the source. For example, if you want to update records in the database before you extract the records from the table, specify a pre-SQL statement.
Post SQL	SQL statement that the Data Integration Service executes after extracting data from the source. For example, if you want to delete some records after the latest records load, specify a post-SQL statement.
Table Name	Overrides the table name of the imported Snowflake source table.

Creating a Snowflake Data Object

Create a Snowflake data object to add to a mapping.

1. Select a project or folder in the **Object Explorer** view.
2. Click **File** > **New** > **Data Object**.
3. Select **Snowflake Data Object** and click **Next**.

   The **Snowflake Data Object** dialog box appears.
4. Enter a name for the data object.
5. Click **Browse** next to the **Location** option and select the target project or folder.
6. Click **Browse** next to the **Connection** option and select the Snowflake connection from which you want to import the Snowflake object.

7. To add a resource, click **Add** next to the **Selected Resources** option.

   The **Add Resource** dialog box appears.
8. Select the checkbox next to the Snowflake object you want to add and click **OK**.
9. Click **Finish**.
   The data object appears under Data Objects in the project or folder in the **Object Explorer** view.

Creating a Snowflake Data Object Operation

You can create the data object read, write, or lookup operation for Snowflake data objects. You can the add the Snowflake data object operation to a mapping.

1. Select the data object in the **Object Explorer** view.
2. Right-click and select **New** > **Data Object Operation**.

   The **Data Object Operation** dialog box appears.
3. Enter a name for the data object operation.
4. Select the type of data object operation. You can choose to create a read or write operation.
5. Click **Add**.

   The **Select Resources** dialog box appears.
6. Select the Snowflake data object for which you want to create the data object operation and click **OK**.
7. Click **Finish**.

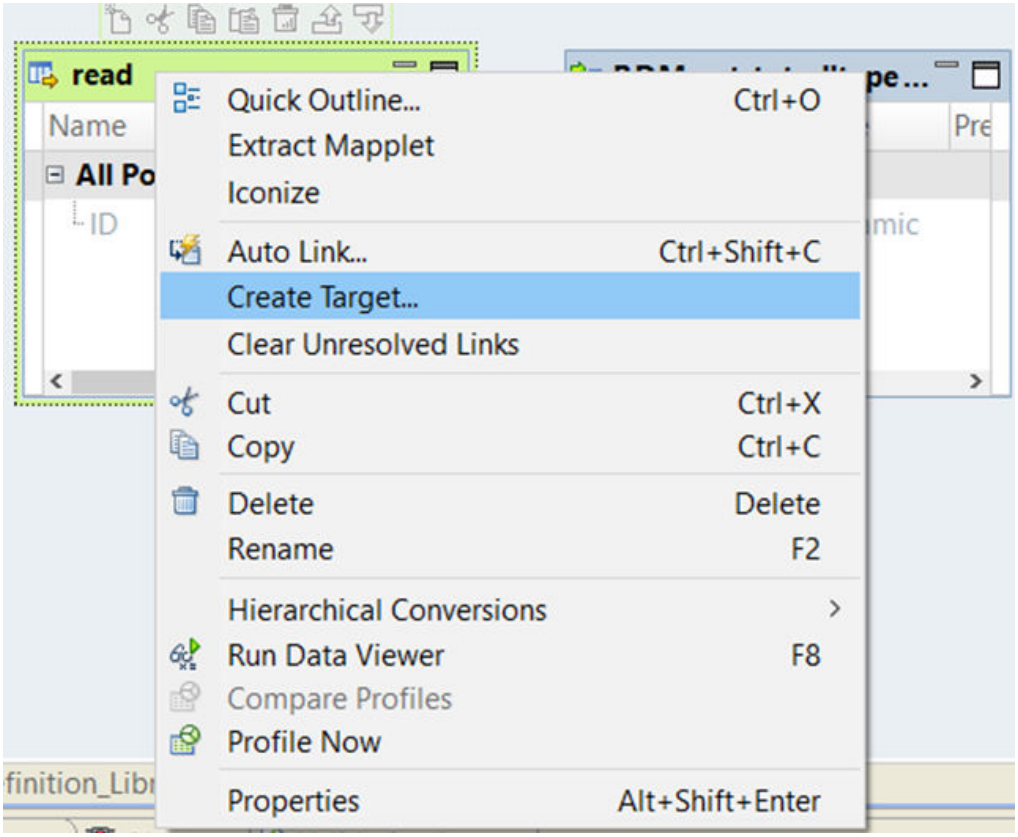
The Developer tool creates the data object operation for the selected data object.

Creating a Snowflake Target

You can create a Snowflake target using the **Create Target** option.

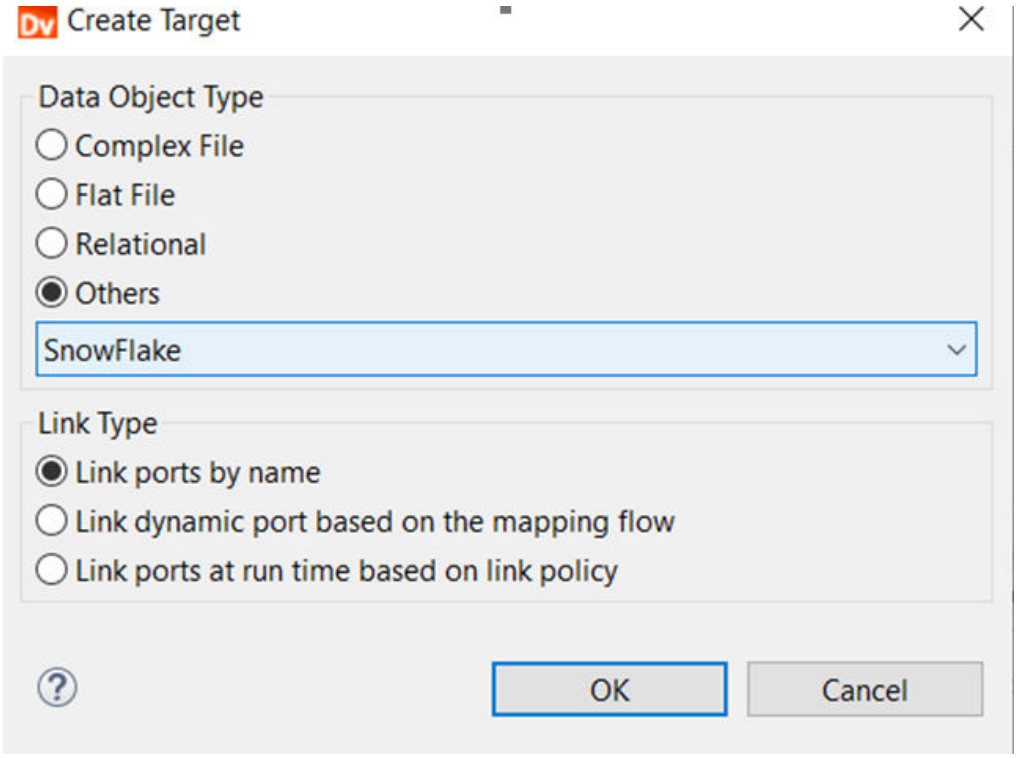
1. Select a project or folder in the **Object Explorer** view.
2. Select a source or a transformation in the mapping.
3. Right-click the Source transformation and select **Create Target**.
   The **Create Target** dialog box appears.

The following image shows the **Create Target** option:



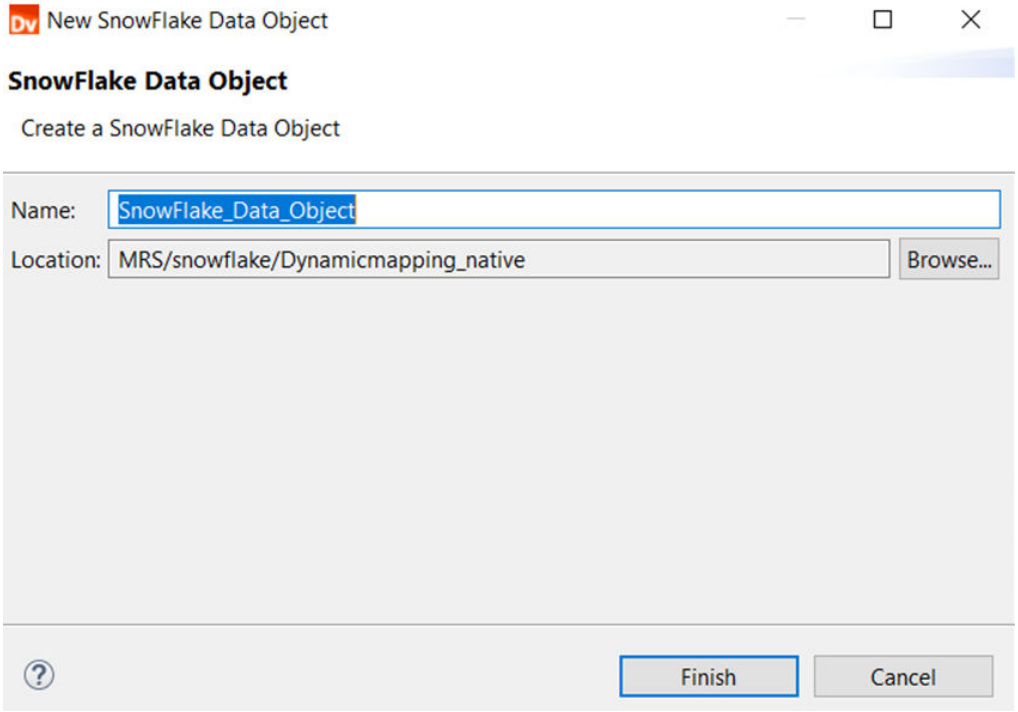
4. Select **Others** and then select **Snowflake** data object from the list in the **Data Object Type** section.

The following image shows the **Data Object Type** section:



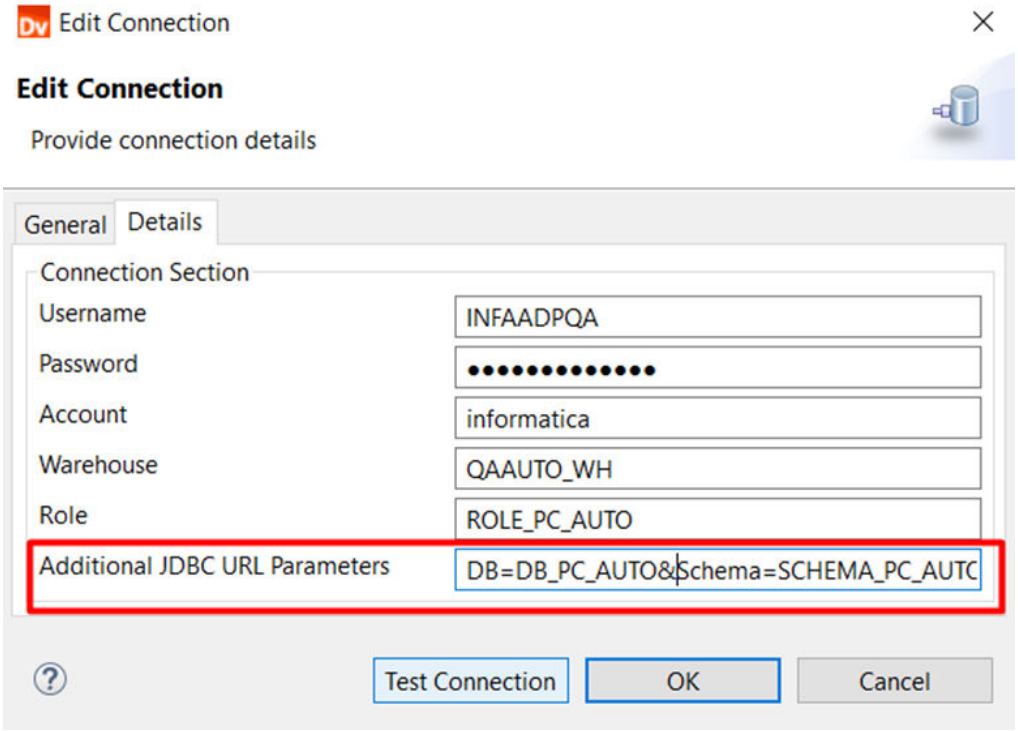
5. Click **OK**.
   The **New Snowflake Data Object** dialog box appears.

   The following image shows the **New Snowflake Data Object** dialog box:

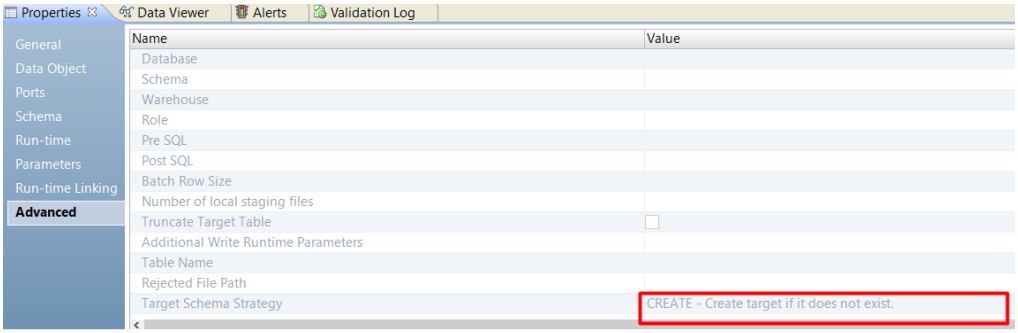


6. Enter a name for the data object.

7. Click **Finish**.
   The new target appears under the **Physical Data Objects** category in the project or folder in the **Object Explorer** view.
8. In the Snowflake connection **Details** tab, specify the database and the schema name to create the target type.
   The following image shows the connection properties:



9. In the Snowflake advanced target properties, select the target schema strategy as **Create - Create target if it does not exist**.
   The following image shows the configured advanced properties:



Adding a Snowflake Data Object Operation as a Snowflake Lookup in a Mapping

Use a Snowflake lookup to look up data in a Snowflake data object.

1. Open a mapping from the **Object Explorer** view.
2. From the **Object Explorer** view, drag a Snowflake data object read operation to the editor.

   The **Add to Mapping** dialog box appears.
3. Select **Lookup** to add the data object read operation as a lookup in the mapping.
4. Click inside the Snowflake object operation and connect the lookup input ports and the lookup output ports.
5. In the **Properties** view, configure the following parameters:
   a. On the **General** tab, select the option that you want the Data Integration Service to return when it finds multiple rows that match the lookup condition.
   b. On the **Lookup** tab, enter the lookup condition properties.
6. When the mapping is valid, click **File** > **Save** to save the mapping to the Model repository.

Rules and Guidelines for Snowflake Mappings

Use the following rules and guidelines when you create a mapping:

- You cannot use the OR operator in a filter condition.
- Temporary tables are not created on Windows when the table name table name contains /\ : * ? " < > | special characters.
- When you configure a native expression to filter Snowflake records where the table name and column name contain special characters, enclose the table name and column name that contain special characters with double quotes in the native expression.
- You must define a primary key in the target table. If you do not define a primary key in the target table, the mapping fails to delete the record from or update the record in the target table.
- When you run mappings in the non-native environment, the Data Integration Service does not consider the JDBC parameters that you specify in the Snowflake connection and the mapping fails.
- When you select the **Retain Target** option with the target schema strategy property in a mapping that runs on Spark engine, and if the target table does not exist, the Integration Service creates the target table and writes the data to the table. A similar mapping that runs in the native environment fails with an error stating that the table does not exist.

CHAPTER 4

PowerExchange for Snowflake Mappings

This chapter includes the following topics:

PowerExchange for Snowflake Mappings Overview

After you create a Snowflake data object read or write operation, you can create a mapping to extract data from a Snowflake source or load data to a Snowflake target.

You can define properties in an operation to determine how the Data Integration Service must extract data from a Snowflake source or load data to a Snowflake target. You can extract data from one or more Snowflake sources, and load data to one or more Snowflake targets. When the Data Integration Service extracts data from the source or loads data to the target, it converts the data based on the data types associated with the source or the target.

Mapping Validation and Run-time Environments

You can validate and run mappings in the native environment or in a non-native environment, such as Hadoop or Databricks.

The Data Integration Service validates whether the mapping can run in the selected environment. You must validate the mapping for an environment before you run the mapping in that environment.

You can configure the mappings to run in the native or non-native environment.

Native environment

When you run mappings in the native environment, the Data Integration Service processes the mapping and runs the mapping from the Developer tool.

Spark Engine

When you select the Hadoop environment, the Data Integration Service pushes the mapping to a compute cluster and processes the mapping on a Spark engine. The Data Integration Service generates an execution plan to run mappings on the Spark engine.

Databricks

When you run mappings in the Databricks environment, the Integration Service pushes the mapping logic to the Databricks Spark engine, the Apache Spark engine packaged for Databricks.

You can view the plan in the Developer tool before you run the mapping and in the Administrator tool after you run the mapping.

For more information about the non-native environment, see the *Data Engineering Administrator Guide*.

Snowflake Mapping Example

Your organization has a large amount of customer data from across regions stored in flat files. You organization needs to analyze data in the APAC region. Create a mapping that reads all the customer records from the flat file and write those records to a Snowflake table.

You can use the following objects in a Snowflake mapping:

Flat file input

The input file is a flat file that contains customer names and their details.

Create a flat file data object. Configure the flat file connection and specify the flat file that contains the customer data as a resource for the data object. Use the data object in a mapping as a read data object.

Transformations

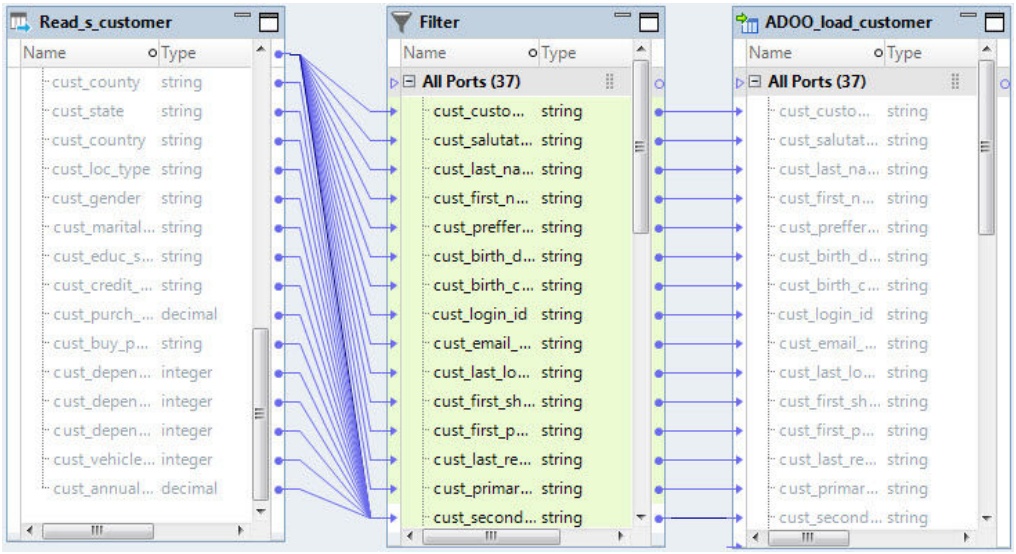
Add a Filter transformation to extract customer data in the APAC region.

The Filter transformation filters the source data based on the value you specify for the region ID column. The Data Integration Service returns the rows that meet the filter condition.

Snowflake output

Create a Snowflake data object write operation. Configure the Snowflake connection and specify the Snowflake object as a target for the data object. Use the data object in a mapping as a target data object.

The following image shows the Snowflake mapping example:



When you run the mapping, the Data Integration Server reads customer records from the flat file and writes to the Snowflake table.

CHAPTER 5

PowerExchange for Snowflake Dynamic Mappings

This chapter includes the following topics:

Snowflake Dynamic Mapping Overview

You can use Snowflake data objects as dynamic sources and targets in a mapping both in the native and the non-native environment.

Use the Snowflake dynamic mapping to accommodate changes to source, target, and transformation logics at run time. You can use a Snowflake dynamic mapping to manage frequent schema or metadata changes or to reuse the mapping logic for data sources with different schemas. Configure rules, parameters, and general transformation properties to create the dynamic mapping.

If the data source for a source or target changes, you can configure a mapping to dynamically get metadata changes at runtime. If a source changes, you can configure the Read transformation to accommodate changes. If a target changes, you can configure the Write transformation accommodate target changes.

You do not need to manually synchronize the data object and update each transformation before you run the mapping again. The Data Integration Service dynamically determine transformation ports, transformation logic in the ports, and the port links within the mapping.

For information about dynamic mappings, see the *Informatica Developer Mapping Guide*.

There are the two options available to enable a mapping to run dynamically. You can select one of the following options to enable dynamic mapping:

- In the **Data Object** tab of the data object read or write operation, select the **At runtime, get data object columns from data source** option when you create a mapping.
  When you enable the dynamic mapping using this option, you can refresh the source and target schemas at runtime.

- In the **Ports** tab of the data object write operation, select the value of the **Columns defined by** property as **Mapping Flow** when you configure the data object write operation properties.

Developing and Running Dynamic Mappings

Perform the following tasks to develop and run a dynamic mapping to read or write to Snowflake. The tasks and the order in which you perform the tasks depend on the mapping scenario and the transformations that you plan to use in the mapping.

1. Create a Snowflake mapping and add the Snowflake objects.
2. Configure a Snowflake dynamic source for the Read or Lookup transformation to get metadata changes from the Snowflake source at run time. Select the Snowflake source object and perform one of the following tasks based on your requirement:
   - Use a parameter as a source for a dynamic mapping source object.
   - Configure data sources for source objects in mappings to get metadata changes at run time. To dynamically get columns from the data source file at run time, select **At run time, get data object columns from the data source**.
3. Create dynamic ports in transformations and link ports.
4. Define input rules for dynamic ports to determine which generated ports to create.
5. Configure a Write transformation to write to a Snowflake dynamic target. Select the Snowflake target object and perform one of the following tasks based on your requirement:
   - Use a parameter as the data object for the transformation and then change the parameter at run time.
   - To dynamically get data object columns from the data source at run-time, enable the option **At run time, get data object columns from the data source**.
   - Define target object columns by mapping flow to enable upstream mapping objects to update the incoming ports for the Write transformation.
     To do this, select **Columns defined by: Mapping flow** in the **Ports** tab of the Properties view and then select **Create or replace table at run time** in the **Target Schema Strategy** list.
6. Create and configure a run-time link to determine which ports to link at run time.
7. Validate the mapping.
8. Compile and run the dynamic mapping.

Snowflake Dynamic Mapping Example

Your organization has a large amount of data that keeps changing. Your organization needs to incorporate all the updated data in a short span of time. Create a dynamic mapping, where you can refresh the source schema dynamically to fetch the updated data. Add all the dynamic ports to the target to override the metadata of the existing ports.

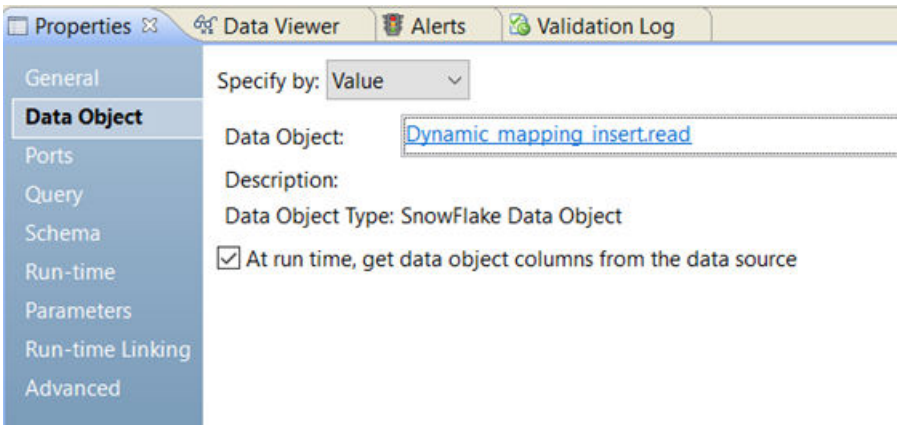
1. Import the Snowflake read and write data objects.
2. Select a project or folder in the **Object Explorer** view.

3. Click **File** > **New** > **Mapping**.

   The **Mapping** dialog box appears.
4. Enter the name of the mapping in the **Name** field.
5. Click **Finish**.
6. Drag the data object into a mapping.

   The **Snowflake Data Object Access** dialog box appears.
7. Select the **Read** option and click **OK**.
8. In the **Data Object** tab, select the **At runtime, get data object columns from the data source** check box.

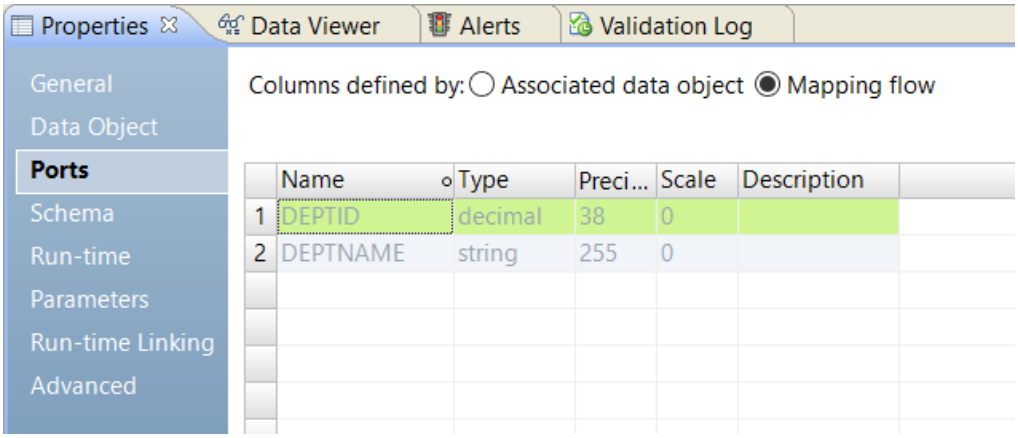
   The following image shows the **Data Object** tab:



9. Drag the data object into a mapping.

   The **Snowflake Data Object Acess** dialog box appears.
10. Select the **Write** option and click **OK**.
11. In the **Ports** tab, select the value of the **Columns defined by** as **Mapping Flow**.

    The following image shows the **Ports** tab:



12. Select all the source incoming ports and add the ports to the target.
13. Save and run the mapping.

Rules and Guidelines for Snowflake Dynamic Mappings

Consider the following rules and guidelines when you configure a dynamic mapping for Snowflake:

- When you configure an update, upsert, or delete operation in a Snowflake dynamic mapping, the Snowflake target table must have the primary key defined for the column. If you configure a mapping flow between the Snowflake source and target in the dynamic mapping, the keys selected in the Snowflake data object are not picked. To rectify this, add a transformation in the mapping. To dynamically get the data object columns from the data source at runtime, select the **At runtime, get data object columns from the data source** check box. Configure the mapping flow option between the source and the transformation, and then enable runtime linking between the transformation and the target.
- Do not add a lookup condition for dynamic ports in a dynamic mapping.

CHAPTER 6

Snowflake Run-Time Processing

This chapter includes the following topics:

Snowflake Run-Time Processing Overview

When you create a Snowflake data object read or write operation, you define properties that determine how the Data Integration Service reads data from or writes data to a Snowflake database.

You can configure lookup caching, partitioning, and parameterization in the run-time properties.

Lookup Cache

You can enable lookup cache for a Snowflake lookup operation. When you enable lookup caching, the Integration Service caches the lookup source and runs the query on all the rows in the cache. The **Run-time** tab displays the **Lookup Caching Enabled** property that the Data Integration Service uses to lookup data from a Snowflake table. Lookup caching is enabled by default.

When you do not configure lookup caching, the Integration Service queries every input row of the lookup source instead of building and querying the lookup cache. You can enable lookup caching for large lookup tables to improve the performance of the lookup operation.

Partitioning

When you read data from or write data to Snowflake, you can configure partitioning to optimize the mapping performance at run time. You can configure partitioning for Snowflake mappings that you run in the native or

Spark engine. The partition type controls how the Data Integration Service distributes data among partitions at partition points.

You can define the partition type as key range partitioning. To configure key range partitioning, open the Snowflake data object read or write operation, and select the **Key Range** partition type option on the **Run-time** tab.

When you configure key range partitioning, the Data Integration Service distributes rows of data based on a port or set of ports that you define as the partition key. You can define a range of values for each port. The Data Integration Service uses the key and ranges to send rows to the appropriate partition.

Note: A Snowflake source mapping configured for key range partitioning does not work in the non-native environment. If you specify a condition for key range partitioning on the Snowflake source and run the mapping on the Spark engine, the Data Integration Service does not append the condition to the select query at run time.

Parameterization for Snowflake Sources

You can parameterize the Snowflake connection and data object read operation properties to override the mapping properties at run time.

You can parameterize the following read operation properties for a Snowflake source:

- Database
- Schema
- Warehouse
- Role
- Pre SQL
- Post SQL
- Table Name

Parameterization for Snowflake Targets

You can parameterize the Snowflake connection and data object write operation properties to override the mapping properties at run time.

You can parameterize the following write operation properties for a Snowflake target:

- Database
- Schema
- Warehouse
- Role
- Pre SQL
- Post SQL
- Table Name
- Batch row size

- Number of local staging files
- Additional write runtime parameters
- Table name

CHAPTER 7

Pushdown Optimization

This chapter includes the following topics:

Accessing Snowflake objects using an ODBC connection overview

You can use pushdown optimization to push transformation logic to Snowflake source databases or target databases. Use pushdown optimization to improve task performance by using the database resources. When you run a task configured for pushdown optimization, the task converts the transformation logic to an SQL query. The task sends the query to the database, and the database executes the query.

You can use an ODBC connection to read from or write large amounts of data to Snowflake. Use source or full pushdown to enhance the performance.

To read data from and write data to a Snowflake object using the ODBC connection, perform the following steps:

1. Download and install the Snowflake ODBC driver.
2. Create a system DSN for Snowflake.
3. Import the Snowflake source and target objects.
4. Create an ODBC connection to access the Snowflake source and target objects.
5. Create and run a mapping.

Install the Snowflake ODBC Driver

Before you establish an ODBC connection to connect to Snowflake, you must install the Snowflake ODBC Driver for Windows. Install the *Snowflake ODBC Driver 32-bit* on the Developer Client machine and the *Snowflake ODBC Driver 64-bit* on the Data Integration Service host machine.

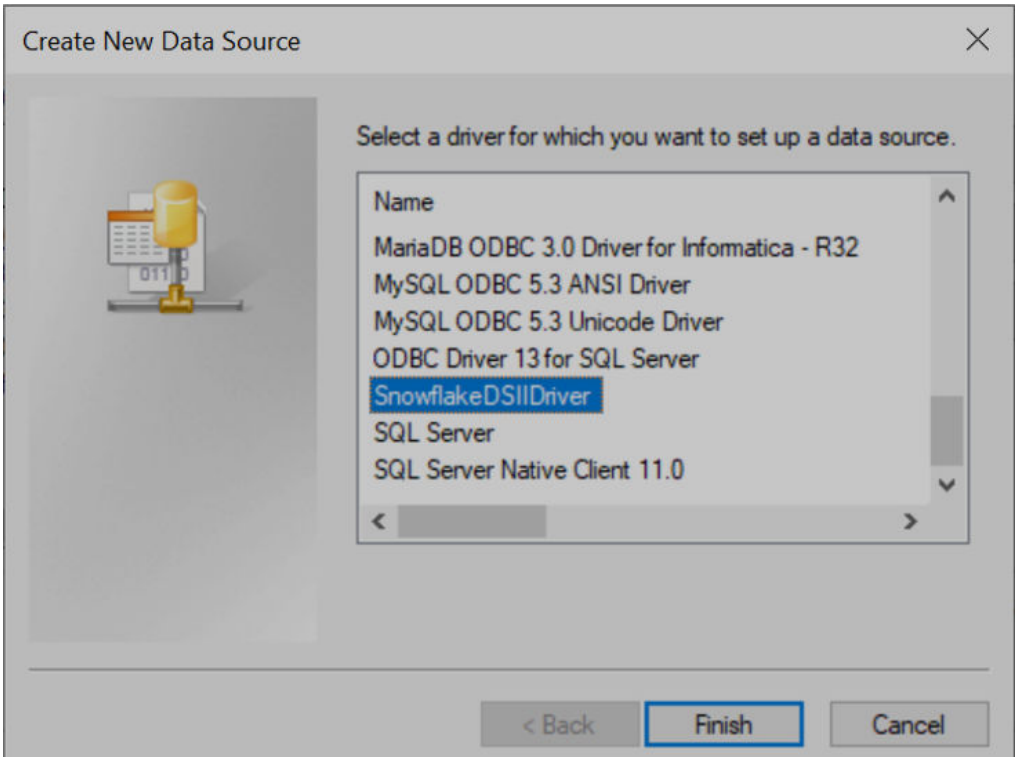
Create a System DSN

Before creating a Snowflake ODBC connection, create a system DSN on the Developer Client machine and the Data Integration Service host machine.

Perform the following steps to create a system DSN on the Developer Client:

1. Open the ODBC Data Sources Administrator (64-bit).
2. Click **System DSN**.
3. Click **Add**.

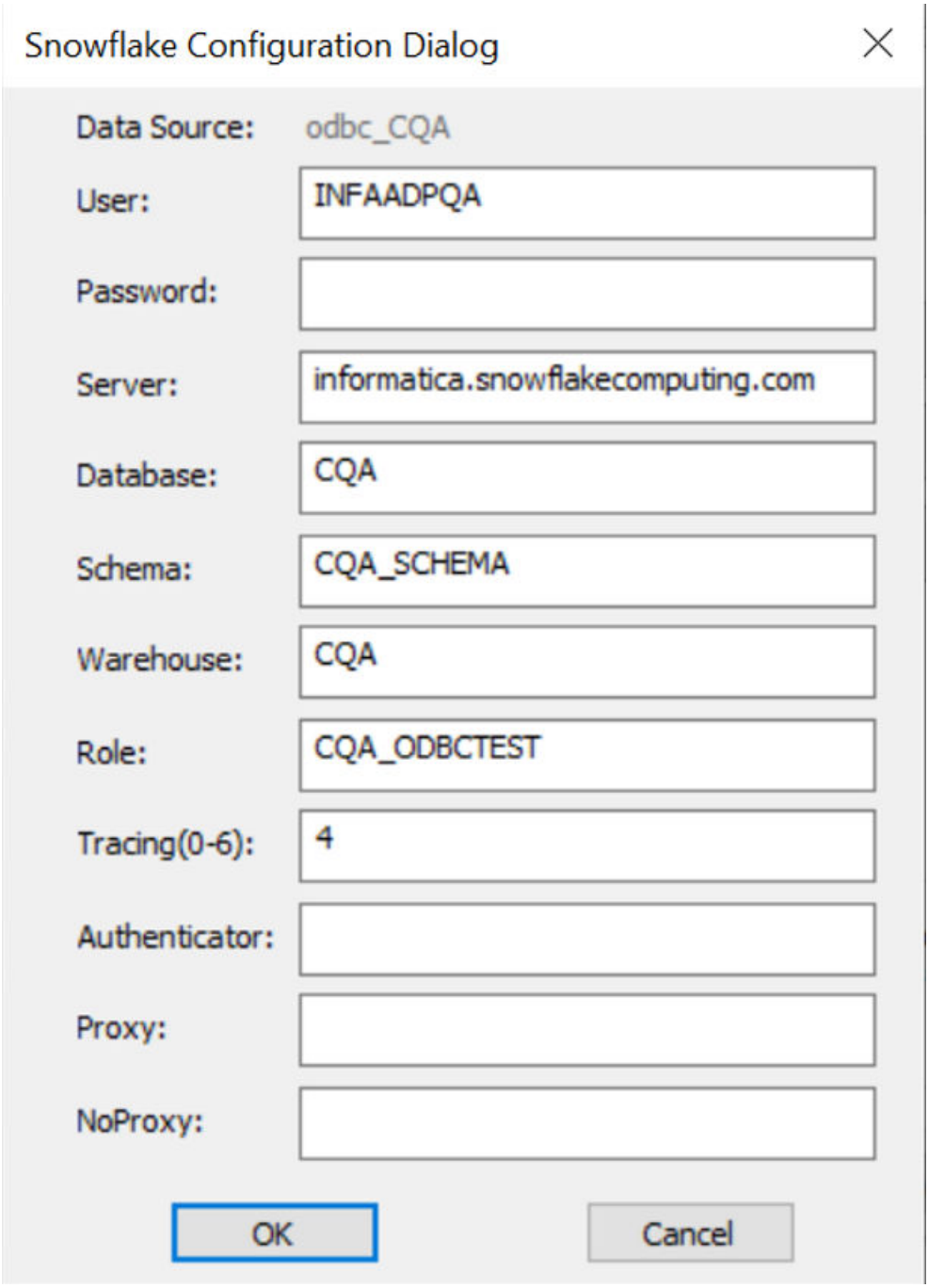
   The **Create New Data Source** dialog box appears.
   The following image shows the **Create New Data Source** dialog box:



4. Select **SnowflakeDSIIDriver** and click **Finish**.

   The following image shows the **Snowflake Configuration** dialog box:

The Snowflake Configuration Dialog box appears.



5. Specify the required details of the Snowflake database that you want to connect to.

   The warehouse name is a mandatory parameter.
6. Click **OK**.

Similarly, you can create a system DSN on the Data Integration Service host machine.

Import the Snowflake Target Object

Import the Snowflake source object from which you want to read data and the target object to write data.

1. Select a project or folder in the Object Explorer view.
2. Click **File** > **New** > **Data Object**.
3. Select **Relational Data Object** and click **Next**.
   The **New Relational Data Object** dialog box appears.
4. Click **Browse** next to the Connection option and select the ODBC connection from which you want to import the Snowflake resources.
5. Click **Create data from existing resource**.
6. To add a resource to the Relational Data Object, click **Browse** next to the Resource option.
7. Navigate to the resource to add it to the data object and click **OK**.
8. Click **Browse** next to the Location option and select the target project or folder.
9. Click **Finish**.

The data object appears under Data Object in the project or folder in the **Object Explorer** view. You can also add resources to a relational data object after you create it.

Create a Snowflake ODBC Connection

Create a Snowflake ODBC connection to access Snowflake source and target objects.

1. In the Developer Client, click **New Connections**.

   The **Select a Connection Category** box appears.
2. Select **Connections** > **Database** > **ODBC**.
3. Create a new Snowflake ODBC connection.
4. Specify the name, description, and location for the connection.
5. Select the **Type** as **Database/ODBC** and click **Next**.
6. Configure the following connection properties:

Connection Property	Description
User Name	The user name to connect to Snowflake.
Password	The password to connect to Snowflake.
Pass Through Security Enabled	Enables pass-through security for the connection. When you enable pass-through security for a connection, the domain uses the client user name and password to log into the corresponding database, instead of the credentials defined in the connection object.
Connection String	The name of the ODBC data source that you created for Snowflake on the Data Integration Service host machine. For example: Snowflake_ODBC_SERVER

Connection Property	Description
Code Page	The code page that the Data Integration Service uses to read or write data.
Environment SQL	SQL commands to set the database environment when you connect to the database. The Data Integration Service runs the connection environment SQL each time it connects to the database.
Transaction SQL	SQL commands to set the database environment when you connect to the database. The Data Integration Service runs the transaction environment SQL at the beginning of each transaction.
SQL Identifier Character	The type of character used to identify special characters and reserved SQL keywords, such as WHERE.
Support Mixed-Case Identifiers	When enabled, the Data Integration Service places identifier characters around table, view, schema, synonym, and column names when generating and executing SQL against these objects in the connection.
ODBC Provider	The type of database to which ODBC connects to perform pushdown optimization. Select the ODBC provider as **Snowflake** to enable the Integration Service to generate native database SQL.

7. Test the connection and click **OK**.

   The Snowflake ODBC connection is created successfully.

Create a Mapping

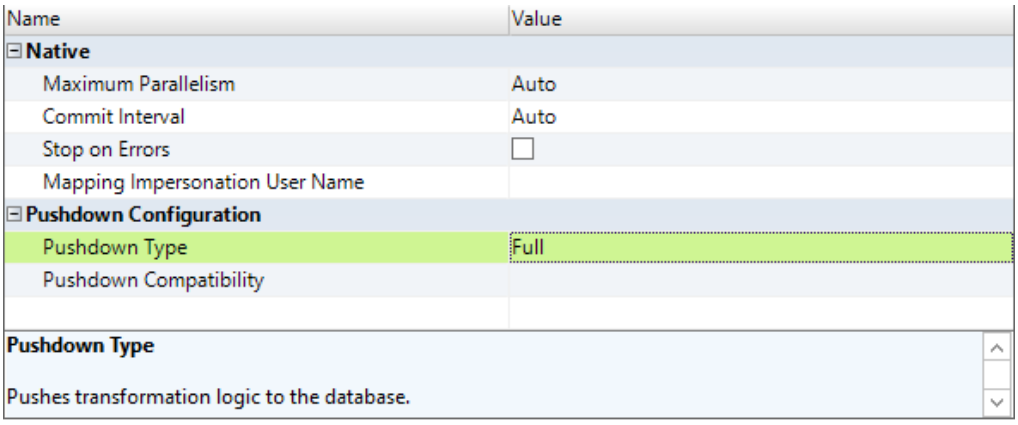
After you import the source and target objects, create a mapping to read data from and write data to Snowflake.

In a mapping, you define properties that determine how the Data Integration Service extracts data from or loads data to a data source. Configure pushdown optimization for the source and target objects.

1. Select a project or folder in the Object Explorer view.
2. Click **Tasks** > **Create**.
3. Click **File** > **New** > **Mapping**.
4. Enter a mapping name and click **Finish**.

   An empty mapping appears in the editor.
5. Drag a data object to the editor and select Read to add the data object as a source.
6. Drag a data object to the editor and select Write to add the data object as a target.
7. On the **Properties** tab, select **Run-time**.
8. Select `Full` as pushdown type.

The following image shows the pushdown optimization configuration:



9. Save and run the mapping.

Pushdown Optimization Functions

PowerExchange for Snowflake supports source and full pushdown optimization.

The following table summarizes the availability of pushdown functions in a Snowflake:

Function	Support
ABORT()	No
ABS()	Yes
ADD_TO_DATE()	Yes
AES_DECRYPT()	No
AES_ENCRYPT()	No
ASCII()	Yes
AVG()	Yes
CEIL()	Yes
CHOOSE()	No
CHR()	Yes
CHRCODE()	Yes
COMPRESS()	No
CONCAT()	Yes
COS()	No

Function	Support
COSH()	Yes
COUNT()	No
CRC32()	No
CREATE_TIMESTAMP_TZ()	No
CUME()	No
DATE_COMPARE()	No
DATE_DIFF()	Yes
DECODE()	Yes
DECODE_BASE64()	No
DECOMPRESS()	No
ENCODE_BASE64()	No
ERROR()	No
EXP()	Yes
FIRST()	Yes
FLOOR()	Yes
FV()	No
GET_DATE_PART()	Yes
GET_TIMESTAMP()	No
GET_TIMEZONE()	No
GREATEST()	No
IIF()	No
IN()	No
INDEXOF()	No
INITCAP()	Yes
INSTR()	No
IS_DATE()	No
IS_NUMBER()	No

Function	Support
IS_SPACES()	No
ISNULL()	No
LAST()	No
LAST_DAY()	No
LEAST()	No
LENGTH()	Yes
LN()	Yes
LOG()	Yes
LOWER()	Yes
LPAD()	Yes
LTRIM()	Yes
MAKE_DATE_TIME()	No
MAX()	Yes
MD5()	No
MEDIAN()	Yes
METAPHONE()	No
MIN()	Yes
MOD()	Yes
MOVINGAVG()	No
MOVINGSUM()	No
NPER()	No
PERCENTILE()	No
PMT()	No
POWER()	Yes
PV()	No
RAND()	No
RATE()	No

Function	Support
REG_EXTRACT()	No
REG_MATCH()	No
REG_REPLACE	No
REPLACECHR()	No
REPLACESTR()	No
REVERSE()	No
ROUND(DATE)	Yes
ROUND(NUMBER)	Yes
RPAD()	Yes
RTRIM()	Yes
SET_DATE_PART()	No
SIGN()	Yes
SIN()	No
SINH()	Yes
SOUNDEX()	No
SQRT()	Yes
STDDEV()	Yes
SUBSTR()	Yes
SUM()	No
SYSTIMESTAMP()	No
TAN()	No
TANH()	Yes
TO_BIGINT	Yes
TO_CHAR(DATE)	Yes
TO_CHAR(NUMBER)	Yes
TO_DATE()	Yes
TO_DECIMAL()	Yes

Function	Support
TO_DECIMAL38()	Yes
TO_FLOAT()	Yes
TO_INTEGER()	Yes
TO_TIMESTAMP_TZ()	No
TRUNC(DATE)	Yes
TRUNC(NUMBER)	Yes
UPPER()	Yes
VARIANCE()	Yes

Pushdown Optimization Transformations

When you configure pushdown optimization for Snowflake, the Data Integration Service pushes the configured transformation to the database.

The Data Integration Service can push the following transformation logic to a Snowflake source or target:

- Filter
- Expression
- Router
- Joiner
- Lookup

APPENDIX A

Snowflake Data Type Reference

This appendix includes the following topics:

Data Type Reference Overview

Developer Tool uses the following data types in Snowflake mappings:

- Snowflake native data types. Snowflake data types appear in Snowflake definitions in a mapping.
- Transformation data types. Set of data types that appear in the transformations. They are internal data types based on ANSI SQL-92 generic data types, which the Data Integration Service uses to move data across platforms. They appear in all transformations in a mapping.

When the Data Integration Service reads source data, it converts the native data types to the comparable transformation data types before transforming the data. When the Data Integration Service writes to a target, it converts the transformation data types to the comparable native data types.

Snowflake and Transformation Data Types

The following table lists the Snowflake data types that Developer Tool supports and the corresponding transformation data types:

Snowflake Data Type	Transformation Data Type	Range and Description
BINARY (VARBINARY)	Binary	Maximum value: 8,388,60 Default value is 8,388,60. **Note:** You can read and write data of Binary data type only in a native environment.
BOOLEAN	String	A Boolean attribute.
DATE	Datetime	Date and time values.

Snowflake Data Type	Transformation Data Type	Range and Description
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, INT, INTEGER, BIGINT, SMALLINT, TINYINT, BYTEINT)	Decimal	Number with 38-bit precision and scale. **Note:** In the native environment, Decimal values only up to 28-bit precision is supported. PowerExchange for Snowflake does not support Decimal values above 28-bit precision for the source or target because of an SDK limitation. However, you can configure the `EnableSDKDecimal38` custom flag in the Data Integration Service properties to read or write data of Decimal data type of 38-bit precision and scale.
TIME	Datetime	Date and time values. **Note:** You can read and write data of Time data type only in a native environment.
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.

INDEX