Snowflake Adapter for SAP Integration Suite

Version 1.2.0 – March 2026

THE BEST RUN SAP

Contents

1 Introduction

1.1 Objective

This is the official guide for the Snowflake Adapter for SAP Integration Suite. This guide covers all relevant information for integration developers to start working with the Snowflake adapter. Read this guide carefully before using the Adapter.

1.2 Coding Samples

Any software coding and/or code lines/strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. The correctness and completeness of the Code given herein are not guaranteed.

1.3 Internet Hyperlinks

The documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. The availability and the correctness of this related information or the ability of this information to serve a particular purpose are not warranted.

1.4 Overview

Snowflake Data Cloud is a cloud-based data platform that provides organizations with a centralized solution for managing, storing, and analysing their data in the cloud.

Snowflake Adapters can be used to implement various data solutions, a few of them are as follows:

- Transaction Analysis for retail business
- Integrating and consolidating data from various sources
- Compute capability for data processing.

1.5 Features

- **Bulk Export and Import:** Supports reading files from multiple input formats like CSV, JSON, and Avro. Export data to your desired format using **unload** operation.
- **Snowflake Operations Support:** Snowflake Adapter offers support for standard Snowflake operations.
- **Multiple options for Staging Storage:** Snowflake Adapter provides various options to load data into **staging** storage like Amazon S3, Google Cloud Storage, or Microsoft Azure.
- **Secure Authentication:** Provides secure authentication with **Database Account** and **Key-Pair** options.
- Bulk Upsert/Unload can work with **Internal and External stage**, **Storage Integration**, and **External storage**.
- Supports **File Staging** operations **(PUT, GET** and **LIST)** to simplify management of data on Snowflake Stage.
- Use **Describe** operation to retrieve data definition information for Snowflake tables.

2 Installation and Configuration

This section details the prerequisites to install and configure the Snowflake adapter.

2.1.1 Prerequisites

> The Snowflake adapter is available as part of your Standard license for SAP Integration Suite. For more information, see SAP Note.

Before you start working with the adapter, you must deploy it to your SAP Integration Suite tenant.

2.1.2 Procedure

> The below installation procedure is compatible with Apache Camel 2, Apache Camel 3, and Edge Integration Cell (EIC) platform.

You can deploy the adapter using the following methods:

2.1.3 Adapter Installation by Creating a New Integration Flow

The Snowflake adapter is available for selection in the receiver adapter list and can be deployed in the **Design** tab directly as you use it in an Integration flow.

Purpose

To install an adapter for use in your Integration flow.

Procedure

Go to **Design** workspace and select the integration package where you want to create a new Integration flow.

1. Click **Edit** to make the package editable.
2. Go to the **Artifacts** tab. Click **Add** and select **Integration Flow**.
3. Enter the **Name** and **ID** for your flow. Additionally, select **Runtime Profile** from the drop-down and choose **Sender** and **Receiver** systems from the list. Finally, click **Add** to create the integration flow.
4. Go to the newly created integration flow and click **Edit** to make it editable.

5. In the integration flow, click **End** to add a **Connector** between the **End** and the **Receiver Box**.
   A drop-down with the available adapters appears. The **Snowflake** adapter should show up in the list.
6. Select the **Snowflake** adapter from the list. The adapter is now imported which triggers an adapter deployment. Once the Snowflake Adapter is deployed, a success message is displayed.

After the above steps are done, the Snowflake Adapter is successfully deployed in your Design workspace of the SAP Integration Suite tenant.

2.1.4 Adapter Installation without Creating a New Integration Flow

The following procedure describes how the Snowflake adapter is migrated from the Discover workspace to the Design workspace of the SAP Integration tenant.

This method is useful for scenarios where integration flow packages are migrated from development to a higher environment such as Production. The Snowflake adapter can be imported into the Design workspace without creating an integration flow. Use the Transport Management Service (TMS) to import/transport the Snowflake adapter to a higher environment. Alternatively, if the TMS is not available in the landscape, the adapter package can be imported into the Design workspace by copying it from the Discover workspace.

Purpose

To import the Snowflake adapter to **Design** workspace by copying the integration package from **Discover** workspace.

Procedure

1. Go to **Discover** workspace.
2. In the search box, search for **Snowflake adapter for SAP Integration Suite** package.
3. Select the package and click **Copy**.
   This copies the package from the Discover workspace to Design workspace.
4. Go to Design workspace and select the copied **Snowflake adapter for SAP Integration Suite** package.
5. In the **Actions** tab of the selected package, click **Deploy**.

   This completes the adapter deployment to Design workspace.

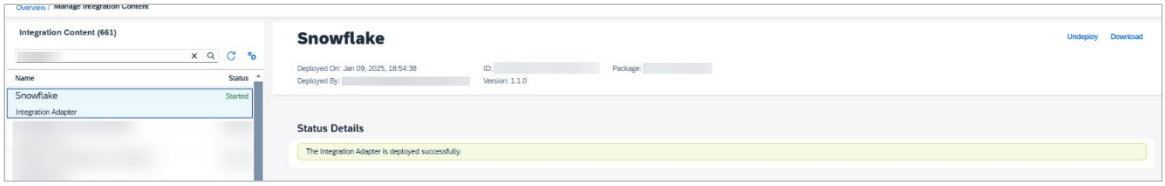
After the adapter deployment is complete, you can check the status in the **Monitor** section.

Purpose

To check the status of the deployed adapter.

Procedure

1. Under the **Monitor** tab, click **Integrations and APIs**. This opens the **Overview** page.
2. On the **Overview** page, go to **Manage Integration Content** section and click **All**. This opens **Integration Content** page with a list of all the deployed adapters.
3. Here, you can check and confirm the deployment status of your adapter.



3 Getting Started: Snowflake Adapter

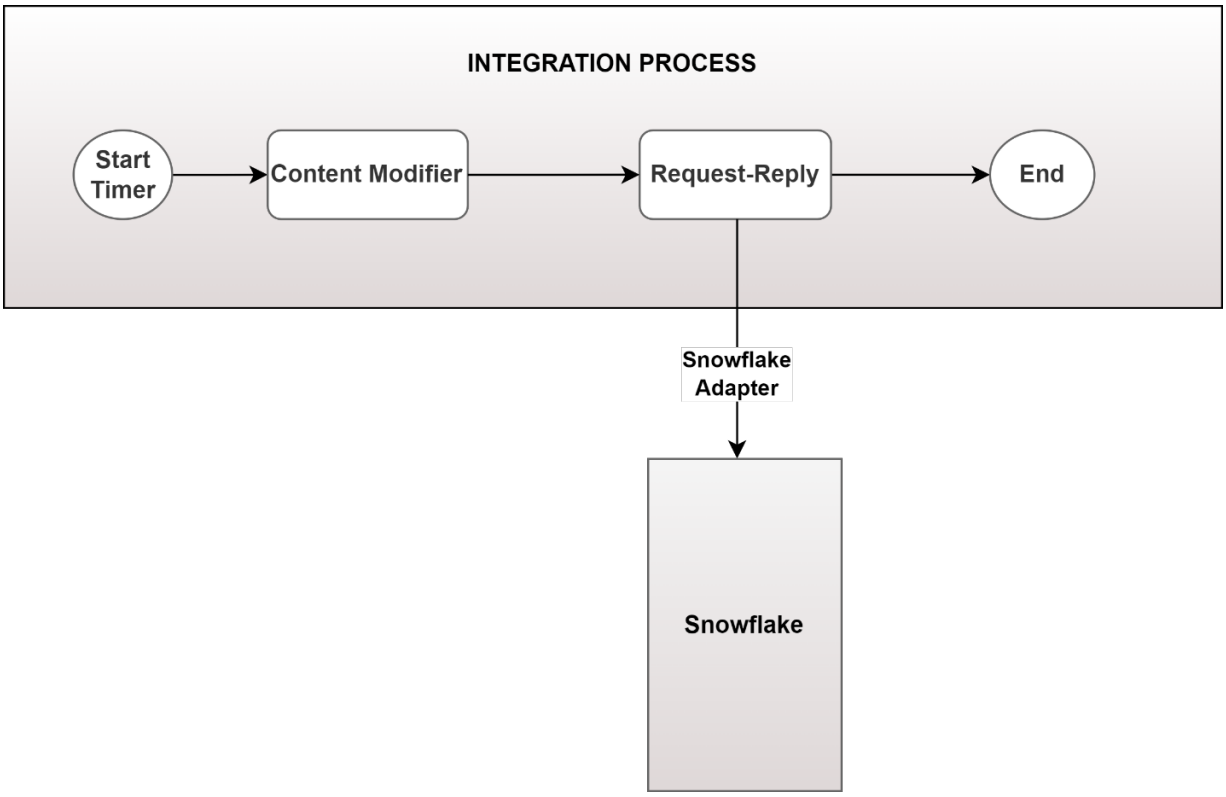
Before you start using the adapter, you can get to know underlying Architecture for the adapter and Application Configuration for Snowflake. Finally, a list of all the Supported Operations have been provided.

3.1 Architecture Overview

The Snowflake adapter is designed to be employed as a receiver adapter. In such a scenario, SAP Cloud Integration acts as the initiator of the calls. The adapter uses database authentication and JDBC as the transport protocol.

You can perform various CRUD operations on your Snowflake application and choose from a list of different staging storage options, such as Amazon S3, Google Cloud Storage, and Microsoft Azure. For more information about the supported operations, see Operations Supported in Snowflake.

The image below gives a high-level representation of how the adapter works. The Snowflake adapter receives database operation information from SAP Cloud Integration and interacts with the Snowflake target application to invoke the same.



3.2 Application Configuration

You can connect to Snowflake via the Snowflake adapter using the same user credentials that you use for Snowflake cloud.

- To start using Snowflake, creating a User Account and generating the user credentials, see Getting Started with Snowflake.
- The accounts created in the Snowflake must have appropriate permissions to use the credentials in the adapter configuration. To manage the user permissions, see Overview of Access Control.

3.3 Authentication

In this section, you will learn about the authentication mechanism supported by Snowflake adapter.

To enable basic authentication for a runtime node, you need to specify user credentials and deploy the attributes. The adapter supports the **Database Account** and **Key Pair** authentication mechanism. For more information see the below sections.

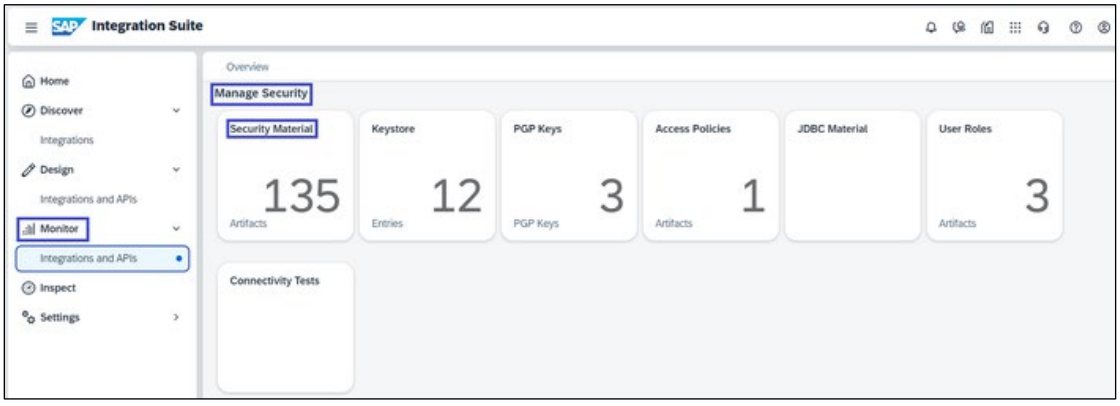
3.3.1 Creating Credentials in Security Material

Purpose

To create credentials in **Security Material** for authentication.

Procedure

1. In SAP Integration Suite, navigate to **Monitor** > **Integrations and APIs**. This opens the **Overview** page.
2. On the **Overview** page, go to **Manage Security** section and click **Security Material**.



3. On **Manage Security Material** page, click **Create** to select **User Credentials** from the dropdown.

4. In the **Create User Credentials** popup, provide the below details.

Parameter	Description
Name	Specify the name for the security artifact. The artifact name is used as an alias for the confidential data assigned by this parameter.
Description	Enter a description for the artifact (optional).
Type	Select `User Credentials`. This allows you to configure a specific system to enable a connection with your integration flow artifact.
User	Specify the username used to invoke the receiver system.
Password	Specify the password against which the user must be authenticated.

5. Click **Deploy** to complete the process.

The Security artifact created above is used to connect to the Snowflake Application by configuring the **Connection tab** of the Adapter.

3.3.2 Creating Keystore for Key-Pair Authentication

Purpose

To create Keystore for **key-pair** authentication.

Prerequisites

1. To use Key-pair authentication, you need to generate a pair of public key and private key (`snowflakersa_key.p8`).
   For more detailed instructions, please refer to the Snowflake Documentation.

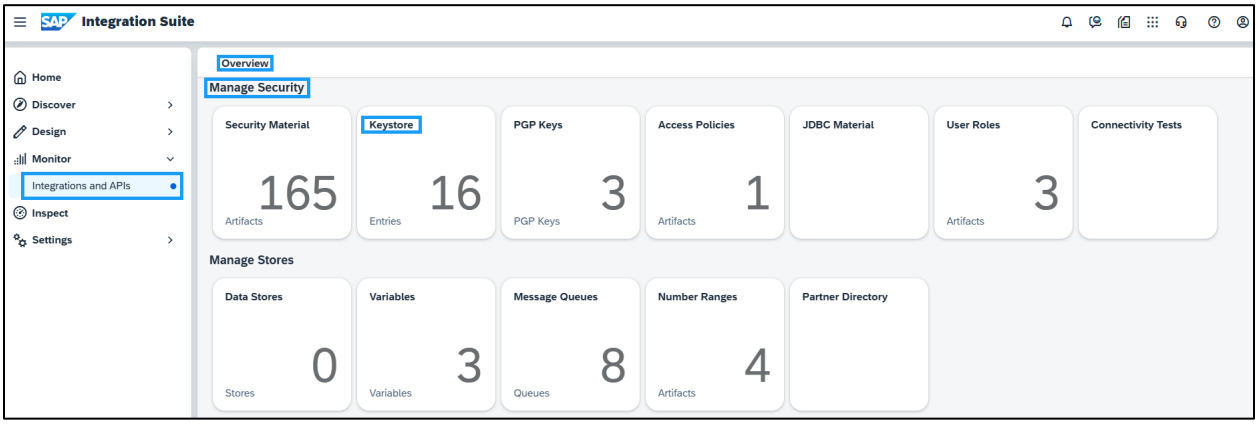
> Ensure to generate multiple active keys to maintain uninterrupted rotation. Rotate and replace your public and private keys based on the expiration schedule. For more detailed information refer Key-Pair Rotation.

2. Now you must link the key to your Snowflake User Account.
   For more information, see Snowflake documentation.

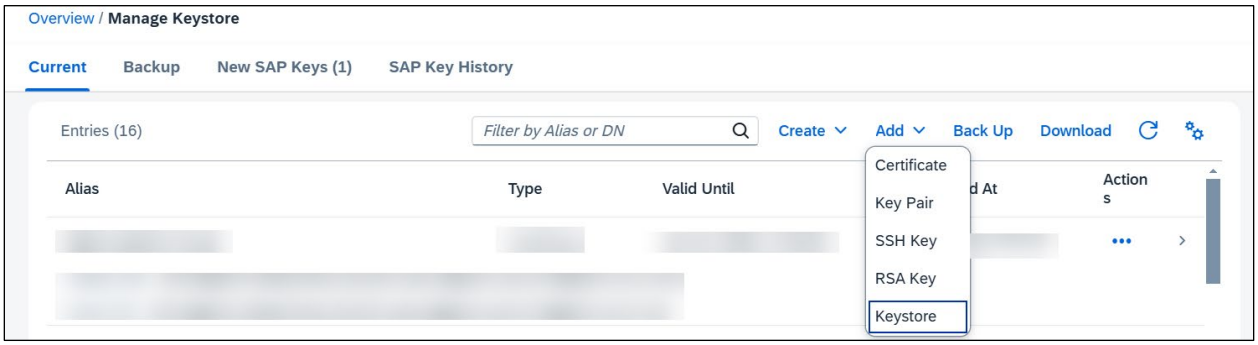
3. Create a keystore file with `.jks` extension.
   For detailed information, see Keystore Generation (JKS file creation).

Procedure

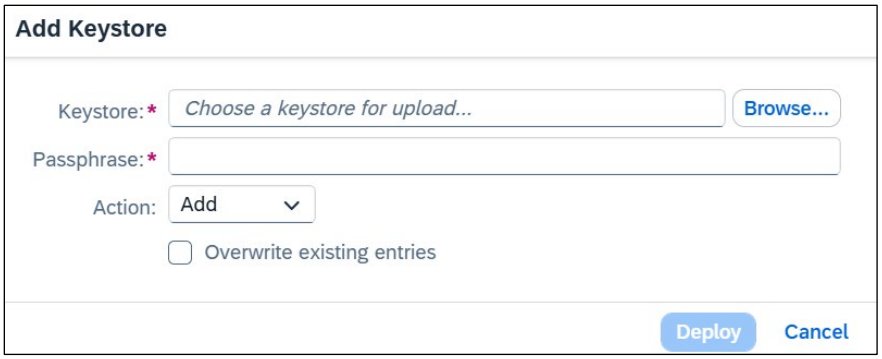
1. In SAP Integration Suite, navigate to **Monitor** > **Integrations and APIs**. This opens the **Overview** page.
2. On the **Overview** page, go to **Manage Security** section and click **Keystore**.



3. On **Manage Keystore** page, click **Add** to select **keystore** from the dropdown.



4. In the **Keystore** popup, provide the below details.



Parameter	Description
Keystore	Choose a keystore for upload. Select the .jks file. For detailed information, see 7.3 Keystore Generation (JKS file creation).
Passphrase	Specify the password used while creating the .jks file.
Action	Specify the Operation to be performed as Add**.**
Overwrite Existing Entries	Enable to overwrite existing entries.

5. Click **Deploy** to complete the process.

3.4 Supported Operations

The Snowflake adapter is based on JDBC driver version 3.17.1. The operations supported by the Snowflake adapter are listed below.

- Bulk Upsert
- Delete
- Execute
- Insert
- Select
- Table List
- Unload
- Update
- Describe
- File Staging

4 Snowflake Adapter Configuration

This section is dedicated to the explanation of configurations and settings for your Snowflake adapter. Specific configurations for each of the Supported Operations are provided. After this section, you can refer to additional information and sample scenarios under References and troubleshooting under Support.

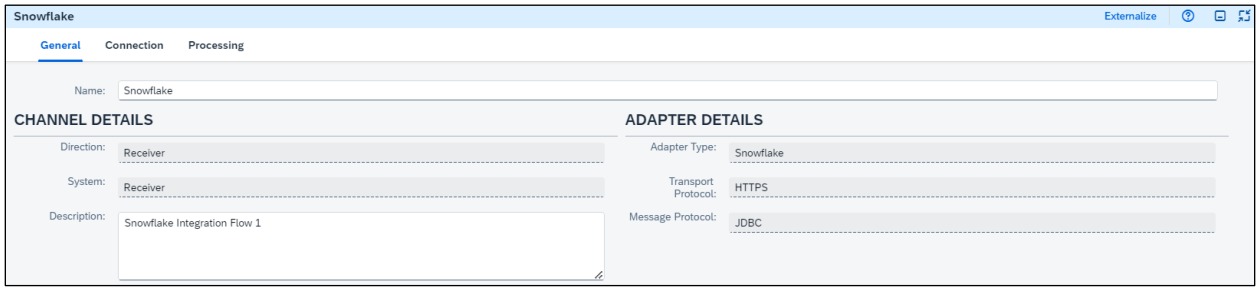
To use this adapter, you will be required to configure the **Connection** and **Processing** tabs. A description and example usage for every field has been added.

4.1 Receiver Adapter

After selecting the Snowflake adapter from the list of adapters, you must configure the **General**, **Connection**, and **Processing** tabs before you can use it.

4.1.1 General

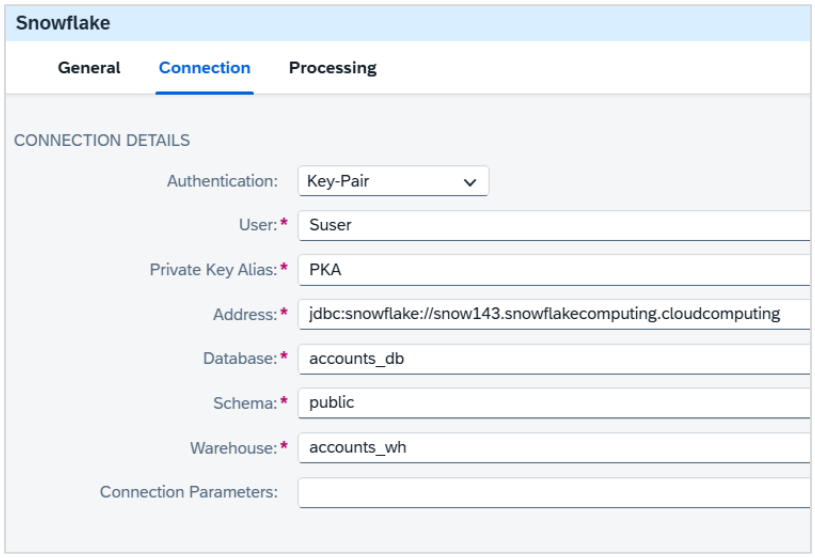
The **General tab** provides an overview of basic adapter information including **Channel** and **Adapter** details.



Parameter	Description
Name	Specify the name of the adapter flow.
Description	Specify the description of the adapter.

4.1.2 Connection Tab

The **Connection tab** contains connection and authentication parameters for Snowflake. Before you set the connection details, see Authentication.



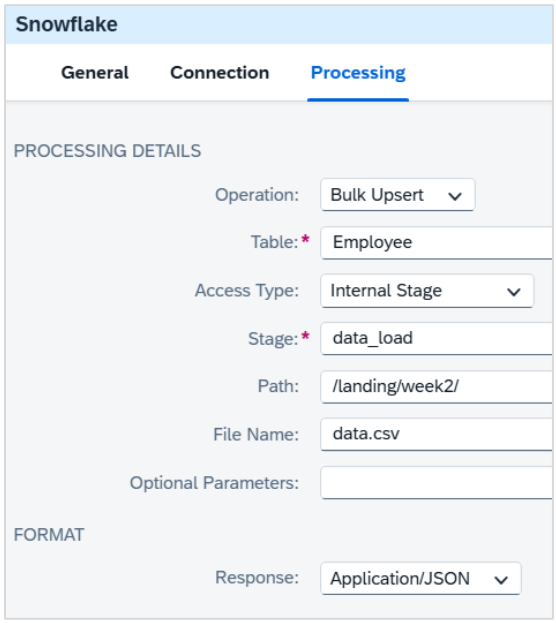
The connection tab contains the following fields:

Parameter	Description
Authentication	Select the authentication mechanism to be used: • **Database Account** • **Key-Pair**
Credential Name	Specify the User Credentials that stores the username/password details in the **Security Material** .
Address	Specify the JDBC endpoint URL of the Snowflake application to be used for the connection. This address typically contains an account identifier which is a combination of the organization and account name separated by a hyphen(`orgname-account_name`). Example: `jdbc:snowflake://<mydb-snow143>.snowflakecomputing.cloudcomputing`

Parameter	Description
Database	Specify the name of the database. Example: `accounts_db`
Schema	Specify the name of the schema to be accessed. Example: `myschema`
Warehouse	Specify the name of the Snowflake warehouse. If the warehouse provided is not found/incorrect, it uses a default warehouse. Example: `accounts_wh` A warehouse is a cluster of computing resources in Snowflake.
Connection Parameter	Specify the optional connection parameters in a query format. If you need to set parameter values that use spaces, ampersands, equals signs, or other special characters, you should URL-encode the special characters. Example `:"?query_tag='folder%3Dfolder1%20folder2%26"`
The below credentials are applicable when **Authentication** is set to **Key-pair**.	
User	Specify the user associated with the Snowflake account.
Private Key Alias	Specify the alias pointing to the Key-pair associated to the user.

4.1.3 Processing Tab

The **Processing tab** contains all the operational configurations for the Snowflake adapter. A sample configuration for the **Bulk Upsert** operation is shown below.



The **Processing** tab contains the following fields:

Parameter	Description
Operation	Specify the type of operation to be executed in Snowflake. For more information, see Supported Operations.
Table	Specify the table on which the operation is to be performed.
Response Fields (Only available when **Operation** is **Select**)	Specify the columns to be returned in response. Separate the fields with a comma. If left empty, the asterisk (*) is assumed by default, and all fields of the table are returned. This is a mandatory field. Example: `'id','firstname','lastname'`

Parameter	Description
Where Clause	Specify "where" clause to be performed on the table. Example: `age >= 25` To delete the entire data, ensure that the condition is set to "**True**" in the **Where Clause** field. This condition applies only to **Delete** operations.
Orderby Statement (Only available when **Operation** is **Select**)	Specify the "order by" clause to sort the result set in ascending or descending order. This is ascending by default. Syntax: `column name DESC/ASC` Example: `first_name DESC`
Limit (Only available when **Operation** is **Select**)	Specify the maximum number of records to be fetched. If no limit is provided all the records will be sent in the response. Example: `50`
Offset (Only available when **Operation** is **Select**)	Specify the offset value for the records to be fetched. Essentially, the number of rows to be ignored before records are retrieved. If an **Offset** is field is to be used in conjunction with **Limit**, otherwise **Offset** value is ignored.
Access Type (Only available when **Operation** is **Unload** or **Bulk Upsert**)	Select the access type for Bulk Upsert/Unload: • **External Stage** references a stage referencing an external cloud storage location. • **External Location** uses an external cloud storage location. • **Internal Stage** references a stage inside your Snowflake environment. • **Storage Integration** is a Snowflake object that stores a generated identity and access management (IAM) entity for your external cloud storage.
Location (Only available when **Access Type** is **External Location**)	Select the location for storing data files (stage) for loading and unloading data.

Parameter	Description
URI (Only available when **Access Type** is **Storage Integration** or **External Location**)	Specify the relative path to be linked to Stage. Example: `s3://staging-bucket/`
Storage Integration (Only available when **Access Type** is **Storage Integration**)	Specify the name of the Storage Integration object in Snowflake.
AWS Key ID Alias (Only available when **Location** is **Amazon S3**)	Specify the alias that stores Access Key to connect to AWS cloud storage.
AWS Secret Key Alias (Only available when **Location** is **Amazon S3**)	Specify the alias that stores Secret Key to connect to AWS cloud storage.
Azure SAS Token Alias (Only available when **Location** is **Microsoft Azure**)	Specify the alias that stores Azure SAS Token to connect to Microsoft Azure cloud storage.
Path	• For **File Stage** operation, specify the path for file staging operation. • For **Unload** or **Bulk Upsert** operation, specify the path which is a prefix for files being referenced in stage.
Stage	Specify the name of internal or external stage in Snowflake. Example: `data_load`

Parameter	Description
File Name	Specify the name of the file. Example: `data.csv` • **Bulk Upsert** operation allows you to specify more than one file. Enclose the filenames in single quotes separated by a comma. Example: `'file.json','file2.json'` • All the file extension must be the same. If left empty, all the files are picked up. • Avoid using this parameter in conjunction with `FILES` option under **Optional Parameters**. Using both will result in two `FILES` values that result in an invalid query.
File Name Prefix (Only available when **Operation** is **Unload**)	Specify the name of the file to be unloaded (without an extension). The extension will be appended to the filename as a suffix depending on the file format of the stage. If you use this in combination with **Optional Parameters** `SINGLE = TRUE`, you can unload your data into a single file, the filename in this case is exactly what you specify here without any additional suffixes.

Parameter	Description
Optional Parameters	Specify one or more optional parameters for **Bulk Upsert/Unload**. • Example for **Bulk Upsert** `FORCE = TRUE MATCH_BY_COLUMN_NAME = CASE_INSENSITIVE FILE_FORMAT = (TYPE = JSON)` For information on valid options usage, see https://docs.snowflake.com/en/sql-reference/sql/copy-into-table • Example for **Unload** Example: `OVERWRITE = TRUE FILE_FORMAT = (TYPE = JSON, COMPRESSION = NONE)` For information on valid options usage, see https://docs.snowflake.com/en/sql-reference/sql/copy-into-location
Stop on record failure (Only available when **Operation** is **Update)**	Enable this if the update operation must stop in case a record fails to update. This field is by enabled by default. The **Update** operation does not throw an error message when a non-existent ID is provided even when **Stop on record failure** is enabled.
SQL Statement (Only available when **Operation** is **Execute**)	Specify the SQL statement to be executed. While using the `Execute` operation for single statement SQL queries, that the query only spans a single line. For multiple queries to be executed as a single statement, specify queries in a single line separated by semicolon. Example: `CREATE TABLE PERSON;DROP TABLE DEPARTMENT;`
Multi-statement execute (Only available when **Operation** is **Execute**)	Enable if the current statement must be executed based on multi-statement execution otherwise single statement execution will be performed.

Parameter	Description
Number of SQL queries (Only available when **Multi-statement execute** is enabled)	Specify the number of SQL queries to be executed. If left at the default value of 0, the number of statements that will be executed is variable.
Schema (Only available when **Operation is Table List**)	Specify the schema containing the tables. If left empty, all the tables from the database are returned. Example: `myschema`
Command	Specify the command for file staging: • **Put** • **Get** • **List**
Response	Specify the response output format: • **Application/JSON** • **Application/XML**

5 Snowflake Adapter Operations

This section contains a comprehensive description of all the operations supported by the Snowflake adapter.

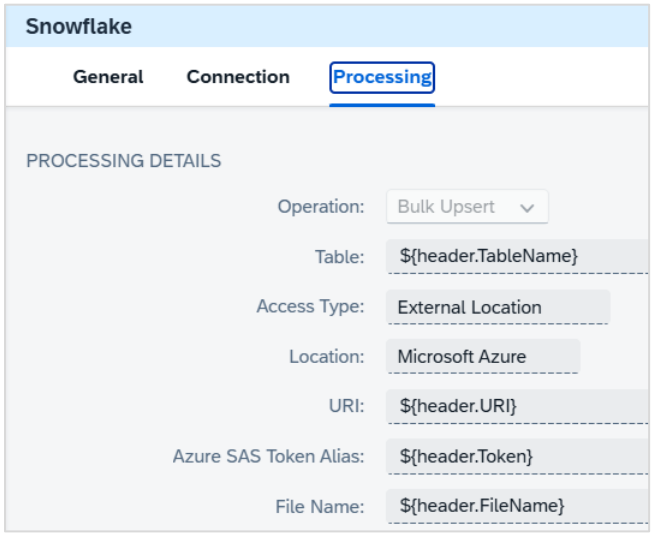
5.1 Bulk Upsert

Bulk Upsert is one of the frequently used operations. You can insert data into Snowflake tables from many different input sources.

5.1.1 External Location

Snowflake allows you to reference an external cloud storage as an input source for data to be inserted in Snowflake tables.

The below screenshot shows an example of Bulk Upsert from a Microsoft Azure storage.



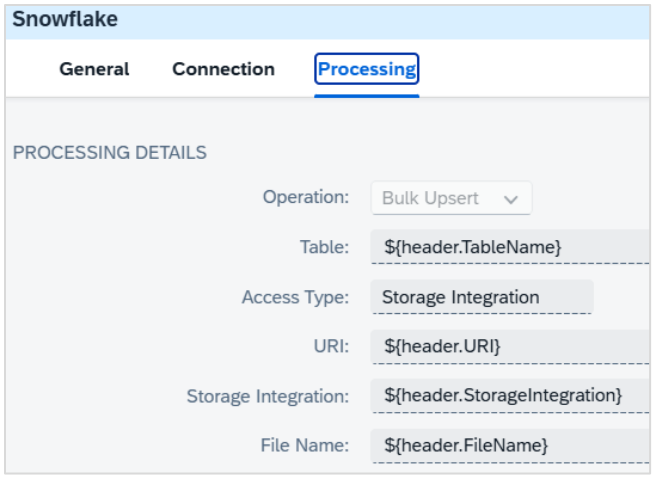
- You must specify the **URI** pointing to the files and folders in Microsoft Azure.
- Also create a **Secure Parameter** artifact that store credentials **Azure SAS** (Shared Access Signature) **Token Alias** to access the external storage.

Parameter	Values
Operation	**Bulk Upsert**
Table	Specify the table to be created in Snowflake as `Customer`
Access Type	Select access type as **External Location**
Location	Select the external cloud storage provider as **Microsoft Azure.**
URI	Specify the relative path to be linked to Stage. `https://org.blob.core.windows.net/container_org/blob_share`
File Name	Specify the name of the files to be stored in the stage. Enclose the filenames in single quotes separated by a comma. `'file.json','file2.json'` • If left empty, all the files are picked from the folder path mentioned in **URI.** • Avoid using this parameter in conjunction with `FILES` option under **Optional Parameters** field. Using both will result in two `FILES` values that result in an invalid query.
Optional Parameters	`FORCE = TRUE MATCH_BY_COLUMN_NAME = CASE_INSENSITIVE FILE_FORMAT = (TYPE = JSON)` For information on valid options usage, see https://docs.snowflake.com/en/sql-reference/sql/copy-into-table

5.1.2 Storage Integration

Storage Integration is a Snowflake object that allows you to access external storage (during Bulk Upsert) of your choice (Microsoft Azure, Google Cloud Storage, Amazon S3) without having to supply any additional credentials.

The below screenshot shows an example of Bulk Upsert using a Storage Integration Object referencing an Amazon S3 bucket.



- You must continue to specify the **URI** to the external cloud (like the **External Location** option).
- You must create a storage integration object associated to the above URI to handle access. For more information, see Create Snowflake Storage Integration.

Sample **Bulk Upsert** Configuration for **Processing** tab:

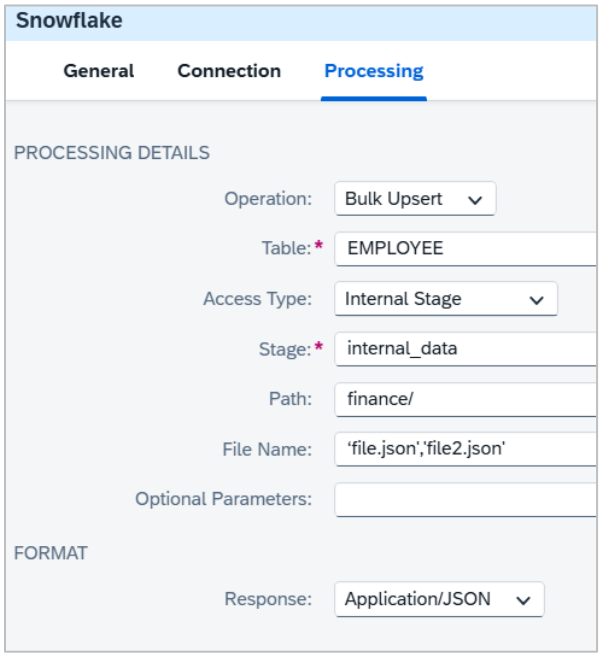
Parameter	Values
Operation	**Bulk Upsert**
Table	Specify the table to be created in Snowflake as `Customer`
Access Type	Select access type as **Storage Integration**
URI	Specify the relative path to be linked to Stage. `s3://bucket-data/datashare/`
Storage Integration	Specify storage integration object as `myobj`
File Name	`'file.json','file2.json'` • If left empty, all the files are picked up from the folder path in URI and a single file is created in Snowflake. • Avoid using this parameter in conjunction with `FILES` option under **Optional Parameters** field. Using both will result in two `FILES` values that result in an invalid query.

5.1.3 Staging

You can use **File Stage** if you wish to specify Snowflake stage as the location for data input. To learn how to manage data on Snowflake File Stage, see 5.10 File Staging.

Internal Stage allows you to perform Bulk Upsert to snowflake in case you do not wish to use an external cloud. To create an internal stage in Snowflake, see Create Stage in Snowflake.

The configuration requires specifying the **target table name** and the **Internal Stage name** from which the data will be read. The **file name** should also be specified, as it determines which file's data will be loaded into the table. If no file name is provided, the adapter will process all files available in the directory.

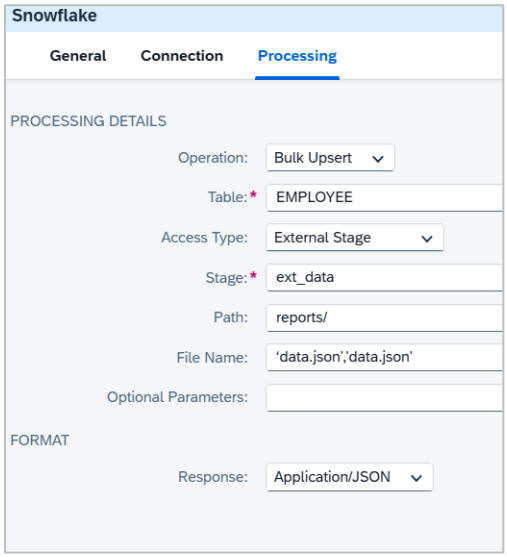


Parameter	Values
Operation	**Bulk Upsert**
Table	Specify the table to be created in Snowflake as `EMPLOYEE`
Access Type	Select access type as **Internal Stage**
Stage	Specify the name of the Stage. `internal_data`

Parameter	Values
Path	Specify the folder path as `finance/`
File Name	Specify file name(s) enclosed in single quotes separated by a comma. `'file.json','file2.json'`

External Stage is a method provided in case you want to link your Snowflake stage so that it stores data on the external cloud provider but can still be accessed using Snowflake Stage.

The configuration is similar to Internal Stage however during external stage creation, you are required to provide external cloud credentials and associate them to your stage. Another benefit of External Stage is that you avoid exposing credentials on SAP CI.

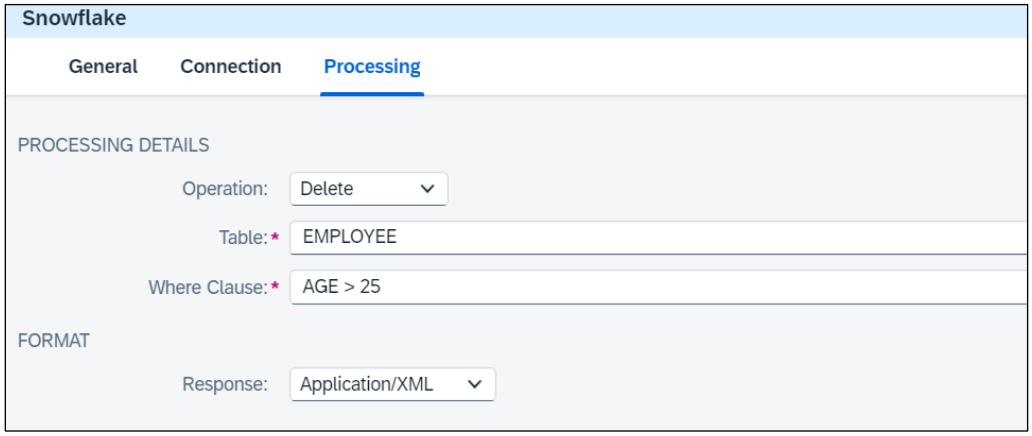


Parameter	Values
Operation	**Bulk Upsert**
Table	Specify the table to be created in Snowflake as EMPLOYEE
Access Type	Select access type as **External Stage**
Stage	Specify the name of the Stage. `ext_data`

Parameter	Values
Path	Specify the folder path as `reports/`
File Name	Specify file name(s) enclosed in single quotes separated by a comma. `'data.json','data.json'`

5.2 Delete

The **Delete** operation deletes rows from the table based on the given condition.



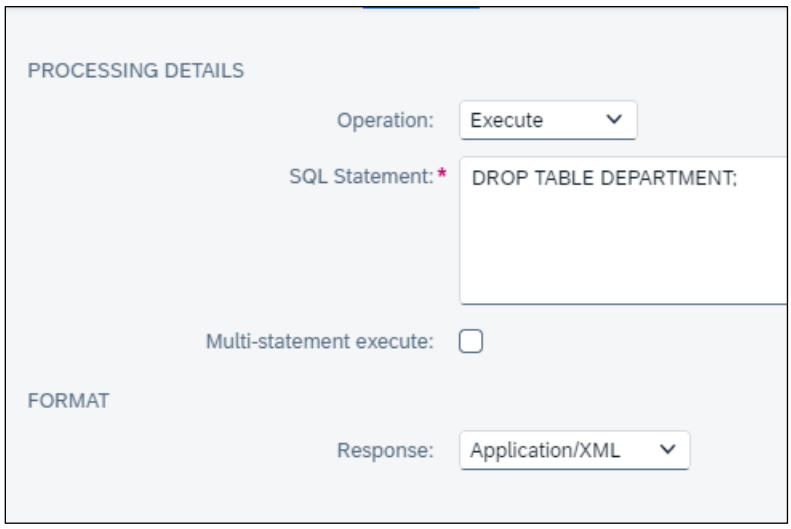
Sample **Delete** Configuration for **Processing** tab:

Parameter	Values
Operation	**Delete**
Table	`EMPLOYEE`
Where Clause	`AGE > 25` To delete the entire data, mention "**True**" in the **Where Clause** field. This field is mandatory.

5.3 Execute

Execute allows you to run SQL queries or stored procedures on the target table.

When EXECUTE is used to perform DDL operation, a successful operation returns "`number of rows affected`" or else an exception is thrown.

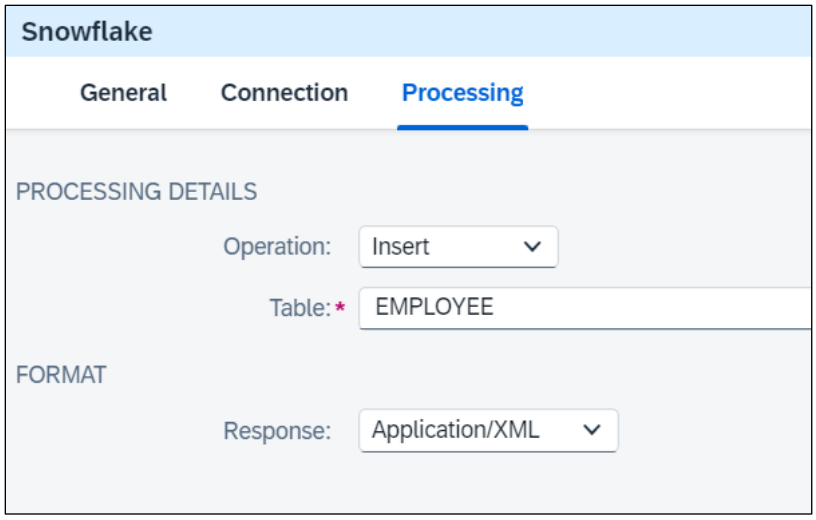


Sample **Execute** Configuration for **Processing** tab:

Parameter	Values
Operation	**Execute**
SQL Statement	`CREATE TABLE PERSON;DROP TABLE DEPARTMENT;` While using the `Execute` operation for single statement SQL queries, ensure the query only spans a single line. For multiple queries to be executed as a single statement, specify queries in a single line separated by semicolon.
Multi-statement execute	Enable this field.
Number of SQL queries	2 This is a mandatory field.

5.4 Insert

Insert operation data into the specified table. You must provide the input data in the payload.



It is mandatory to include `datatype` attribute for `fieldname` in the INSERT payload. metadata/row tags are case sensitive and must be in lower case.

```
<root>
	<metadata>
		<fieldname datatype="NUMBER">ID</fieldname>
		<fieldname datatype="VARCHAR">FIRST_NAME</fieldname>
		<fieldname datatype="VARCHAR">LAST_NAME</fieldname>
		<fieldname datatype="VARIANT">EMAIL</fieldname>
	</metadata>
	<row>
		<ID>10002</ID>
		<FIRST_NAME>Mark</FIRST_NAME>
		<LAST_NAME>Adams<LAST_NAME>
		<EMAIL>{"subject":"Message","emailContent":"Old Content"}<EMAIL>
	</row>
	 <row>
		<ID>10003</ID>
		<FIRST_NAME>James</FIRST_NAME>
		<LAST_NAME>Castor<LAST_NAME>
		<EMAIL>{"subject":"Message","emailContent":"New Content"}<EMAIL>
	</row>
</root>
```

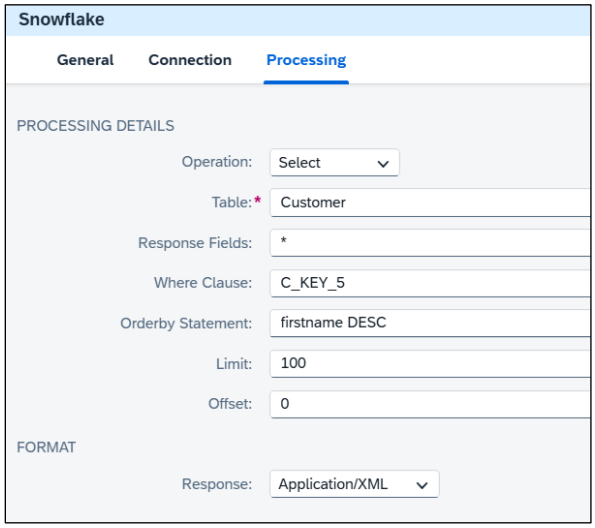
Sample **Insert** Configuration for **Processing** tab:

Parameter	Values
Operation	**Insert**
Table	`Customer`

5.5 Select

Select is used to fetch data from the table. You can customize data by selecting required columns in **Response Fields** for a specific condition using **Where Clause**. You can use **Limit** to fetch a limited number of records from a predefined **Offset** value and finally, **Orderby Statement** allows you to return records in descending or ascending order.

The configuration details are as follows:



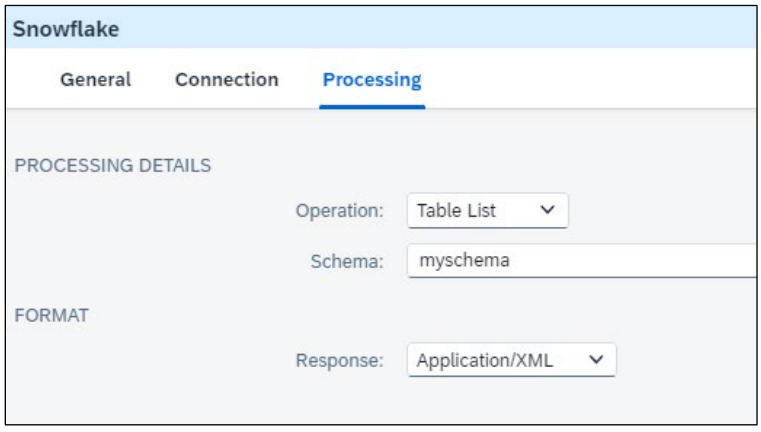
Sample **Select** Configuration for **Processing** tab:

Parameter	Values
Operation	**Select**
Table	`Customer`

Parameter	Values
Response Fields	`id,firstname,lastname` If left empty, the asterisk (*) is assumed by default, and all fields of the table are returned.
Where Clause	`C_KEY > 5`
Orderby Statement	`firstname DESC`
Limit	100 If no limit is provided all the records will be sent on the response.
Offset	0 If an offset is provided, **Limit** cannot be empty.

5.6 Table List

Table List operation lists the tables for the current specified schema.



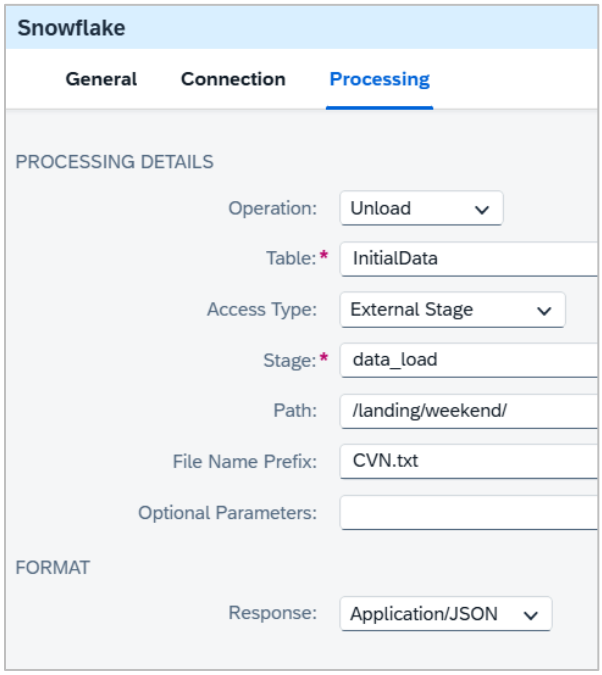
Sample **Table List** Configuration for **Processing** tab:

Parameter	Values
Operation	**Table List**

Parameter	Values
Schema	`Myschema` If no schema is specified, all the tables from the database will be returned.

5.7 Unload

Unload operation exports the data from tables into flat or delimited text files.



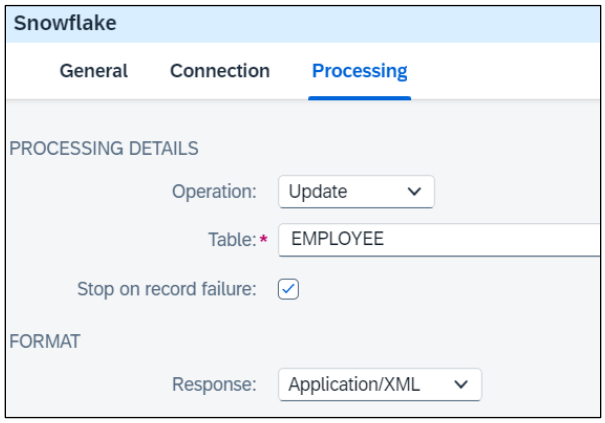
Sample **Unload** Configuration for **Processing** tab:

Parameter	Values
Operation	Select the operation as **Unload**
Table	Specify the table to be created in Snowflake as `InitialData`
Access Type	Select the access type option as **External Stage**
Stage	Select the name of the stage as `data_load`.
Path	Specify the path in the stage as `landing/week1/`

Parameter	Values
File Name Prefix	Specify the name of the file to be unloaded (without an extension). The extension will be appended to the filename as a suffix depending on the file format of the stage. Example: `data` If you use this in combination with **Optional Parameters** `SINGLE = TRUE`, you can unload your data into a single file, the filename in this case is exactly what you specify here without any additional suffixes (without any extension).
Optional Parameters	`OVERWRITE = TRUE FILE_FORMAT = (TYPE = JSON, COMPRESSION = NONE)` For information on valid options usage, see https://docs.snowflake.com/en/sql-reference/sql/copy-into-location.

5.8 Update

The **Update** operation allows you to perform updates on existing data.



You must provide where clause data in the payload as shown below.

- metadata/row tags are case sensitive and must be in lower case.
- For update, ensure metadata fieldname exists for your row data entry.
  - In case your payload does not contain row entry for a metadata fieldname, update will still work.

- However, the opposite is not true, without specifying metadata fieldname for a column, you cannot update a row entry.

```
<root>
	<metadata>
		<fieldname datatype="NUMBER">ID</fieldname>
		<fieldname datatype="VARCHAR">FIRST_NAME</fieldname>
		<fieldname datatype="VARCHAR">LAST_NAME</fieldname>
		<fieldname datatype="VARIANT">EMAIL</fieldname>
	</metadata>
	<row>
		<ID>10002</ID>
		<LAST_NAME>Adams<LAST_NAME>
		<EMAIL>{"subject":"Message","emailContent":"New Content"}<EMAIL>
		<where>ID=10001</where>
	</row>
</root>
```

Sample **Update** Configuration for **Processing** tab:

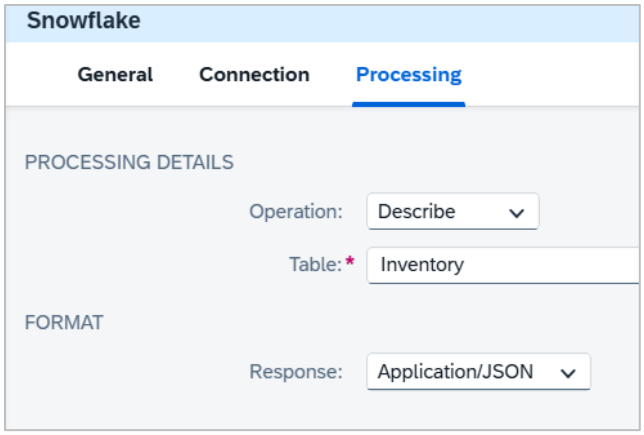
Parameter	Values
Operation	Specify operation as **Update**
Table	Specify Table as `EMPLOYEE`
Stop on record failure	Enable to continue processing even if an error occurs during record update.

`Update` does not throw an error message when a non-existent ID is provided even when **Stop on record failure** is enabled.

5.9 Describe

Describe allows you retrieve the details for a specific object. For more information see, Snowflake SQL Reference.

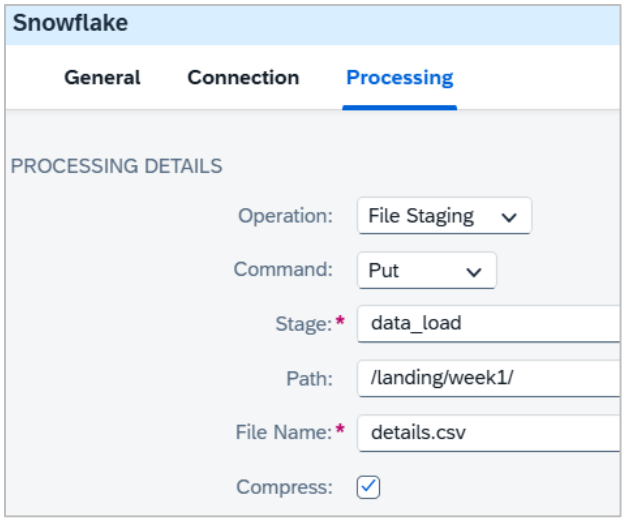
The below screenshot shows the configuration for **Describe** operation on a table.



Parameter	Values
Operation	Specify operation as **Describe**
Table	Specify Table as `Inventory`
Response	Select the response type. Describe operation on a table only supports **Application/JSON** response format.

5.10 File Staging

File Staging operation allows you to manage files on the internal stage. You can upload (Put), download (Get) or List files using this operation.



Parameter	Values
Operation	Specify operation as **File Staging**
Command	Specify command as **Put** A Put operation always overwrites any existing files in the target stage with the local files being uploaded.
Stage	Specify the name of your internal stage as `data_load.`
Path	Specify the folder path in stage where fille must be placed. `landing/week1`
File Name	Specify the name of the file. `details.csv`
Compress	Enable to compress size during file upload to stage.

6 Support

6.1 Tips

- You can monitor, debug, and analyze errors or issues by changing the **Log level** of your integration flow to **Traces**. For more information, see Tracing.
- Before sending the operation for execution, Snowflake adapter generates an SQL query that is also stored in the exchange property `Snowflake_Query`. This information can be handy for analysing your query.
- When EXECUTE is used to perform DDL operation, a successful operation returns "`number of rows affected`" or else an exception is thrown.

6.2 Troubleshooting

A few issues that you might encounter and possible solutions for them have been documented below.

Error	Error Message	Possible Solution
Unsuccessful Connection Attempt with An Invalid Account Name	`JDBC Driver encountered communication error. Message: HTTP status=403`	You must validate your account and credentials before making a connection attempt.
Connecting to a non-existent or mistyped database	`Object '<table_name> does not exist or not authorized.`	Adapter avoids throwing custom error messages since the Snowflake application already flags the error. Ensure account, database and warehouse details are validated before making a connection attempt.
Error Message for Connection Attempts with Invalid Warehouse	`org.apache.camelException: No active warehouse selected in the current session. Select an active warehouse with the 'use warehouse' command.`	Ensure the warehouse name is correct or select an active warehouse for your session.

Error	Error Message	Possible Solution
Adapter retrieves data when the schema is Invalid or empty but throws an error in case the PUBLIC schema is dropped.	`org.apache.camelException: SQL Compilation Error: Object 'SIMPLETABLE' does not exist or not authorized.`	Ensure the schema name is correct. By default, schema points to PUBLIC if schema name is empty or incorrect.

The Snowflake Adapter Fails to Execute Single Statement SQL Query spanning over multiple lines.	`[CONTENT] [CONTENT_DEPLOY] [InstanceError] : {"message":"EXCEPTION", "parameters":["org.apache.camel. FailedToCreateRouteException: Failed to create route Process_1`
	
	...<error snippet>...
	
	`because of Failed to resolve endpoint: Illegal`
	.....<error snippet>.....
	
	`Illegal character in opaque part at index 162:`
		It is recommended that you do not have any line separators in your query while using "Execute".

Incorrect user during Key-Pair authentication.	`SQL error occurred: JWT token is invalid. [f54e3805-e712-4427-aab2-f995e0c3486d].`	The username and password must be checked whenever this exception is thrown.

7 References

7.1 Configuring Secure Access

For `Bulk Upsert` and `Unload` operations, configure secure access from Snowflake to Amazon S3 Bucket. There are two ways this can be done:

- Configure Snowflake Storage Integration Object. For more information, see Stage Configuration.
- Configure AWS IAM Credentials. For more information, Configuring AWS IAM.

Similarly, Bulk Loading from Google Cloud Storage and Microsoft Azure can be performed after configuring relevant access. For more information see, Bulk Loading from GCS and Bulk Load from Azure.

7.2 Snowflake Data Types

For Snowflake operations, ensure your data is aligned with the recommended usage guidelines for **Snowflake Datatypes**. For more information, see Snowflake Data Types.

7.3 Keystore Generation (JKS file creation)

To create Security Certificate using OpenSSL, follow the steps below:

1. Generate a certificate signing request by using the `snowflakersa_key.p8` file and store it in a file called `snowflakeserver.csr`.

   OpenSSL Command:

   ```
   openssl req -new -key <source_path>/snowflakersa_key.p8 -out
   <target_path>/snowflakeserver.csr
   ```

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [AU]:<enter_country_code>
State or Province Name (full name) [Some-State]:<enter_city_name>
Locality Name (eg, city) []:<enter_locality_name>
Organization Name (eg, company) [Internet Widgits Pty
Ltd]:<enter_org_name>
Organizational Unit Name (eg, section) []:<enter_org_unit_name>
Common Name (e.g. server FQDN or YOUR name) []:<enter_common_name>
Email Address []:<enter_email_address>

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:<password>
An optional company name []:<company_name>
```

2. It is possible to sign the certificate using a self-signing mechanism or via CA.

> Please refer to your organisation policy to choose a suitable signing mechanism for the certificate.

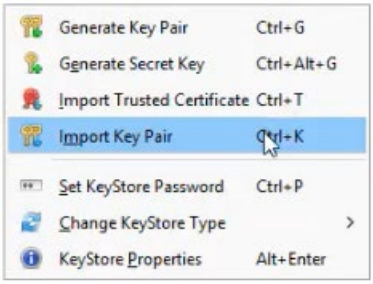
You can generate a self-signed digital certificate using our private key (`snowflakersa_key.p8`) and certificate signing request (`snowflakeserver.csr`) file and store the certificate in a file called (`snowflakeserver.crt`).

This `snowflakeserver.crt` is uploaded when creating the jks file.

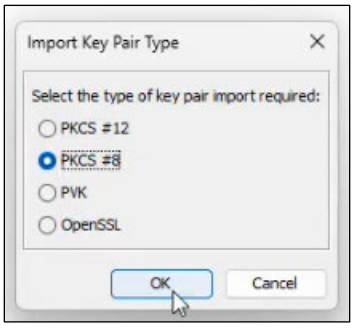
OpenSSL Command:

```
openssl x509 -req -sha256 -days 365 -in
<source_path>/snowflakeserver.csr -signkey
<source_path>/snowflakersa_key.p8 -out
<target/path>/snowflakeserver.crt
```

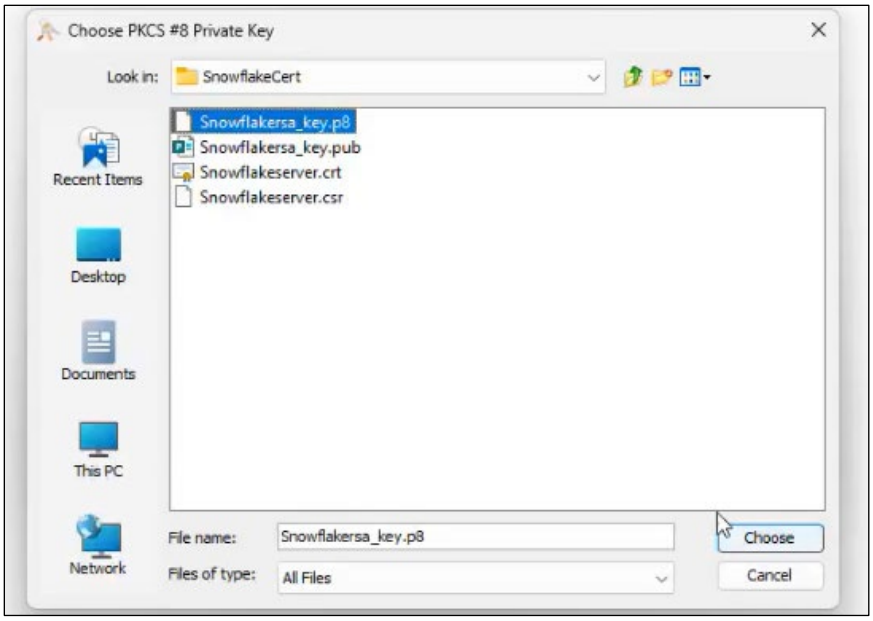
3. To create a JKS file using Keystore in Windows follow the below steps:
   i. Go to KeyStore and **Import key pair**.



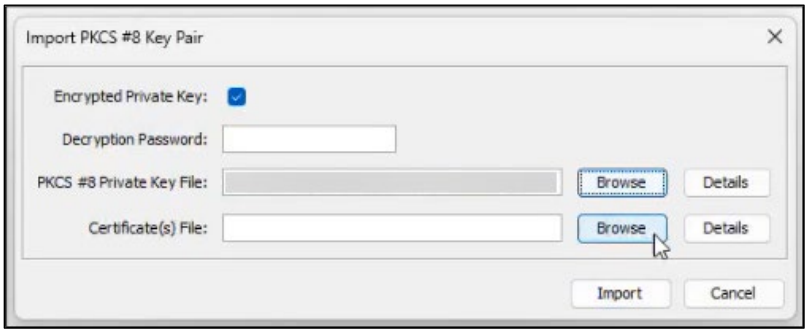
ii. Select the type of key pair as **PCKS #8**.

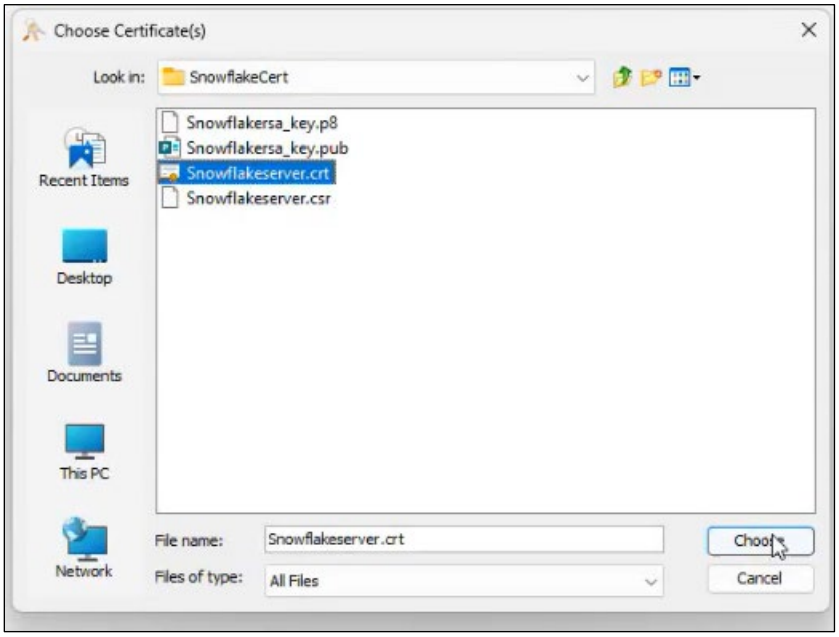


iii. Add the private key file by uploading the **Private Key File**.

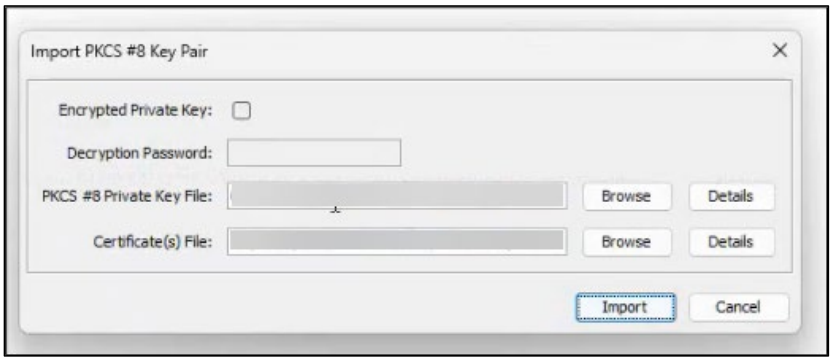


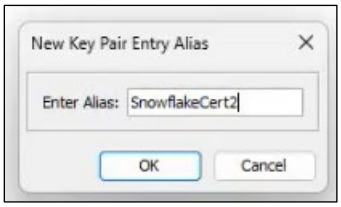
iv. Add the snowflake **Certificate** created above.



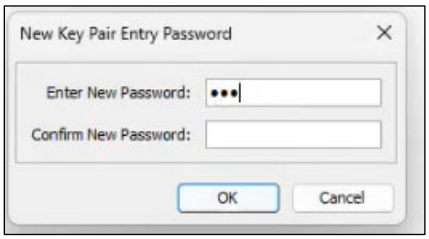


v. Click **Import** and provide the **Alias**.



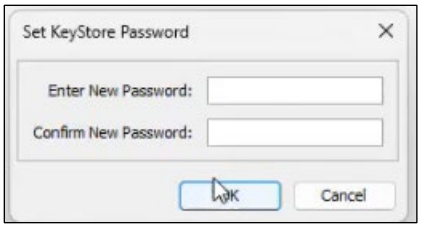


vi. Provide a **New Key Pair Entry Password**.



vii. You will get a message indicating key pair import was successful

viii. Provide the **Key Store Password**.



ix. Finally, save the .jks file.