Requirements for Snowflake

Since v5.2.3/16

Contents

	Snowflake	
Capture	Hub	Integrate
✗	✗	✓

This section describes the requirements, access privileges, and other features of HVR when using Snowflake for replication. For information about compatibility and supported versions of Snowflake with HVR platforms, see Platform Compatibility Matrix.

For the Capabilities supported by HVR on Snowflake, see Capabilities for Snowflake.

For information about the supported data types and mapping of data types in source DBMS to the corresponding data types in target DBMS or file format, see Data Type Mapping.

For instructions to quickly setup replication using Snowflake, see Quick Start for HVR - Snowflake.

ODBC Connection

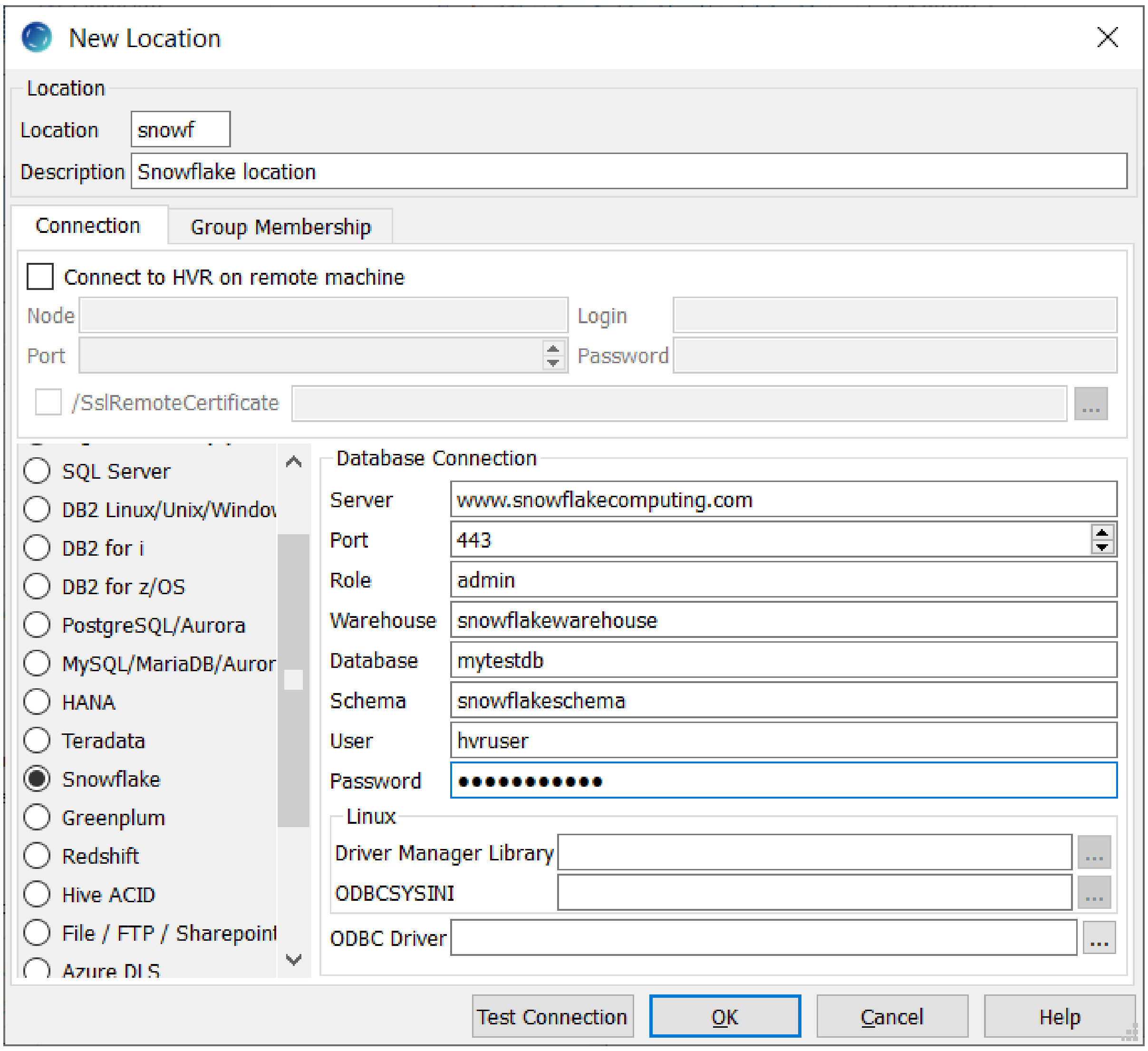
HVR requires that the Snowflake ODBC driver is installed on the machine from which HVR connects to Snowflake. For more information on downloading and installing Snowflake ODBC driver, see Snowflake Documentation.

For information about the supported ODBC driver version, refer to the HVR release notes (**hvr.rel**) available in **hvr_home** directory or the download page.

After installing the Snowflake ODBC driver, configure the **LogLevel** configuration parameter as specified in ODBC Configuration and Connection Parameters of the Snowflake Documentation.

Location Connection

This section lists and describes the connection details required for creating Snowflake location in HVR.



New Location

Location

Location: snowf

Description: Snowflake location

[ ] Connect to HVR on remote machine

Node | Login | Port | Password | /SslRemoteCertificate

SQL Server / DB2 Linux/Unix/Windov / DB2 for i / DB2 for z/OS / PostgreSQL/Aurora / MySQL/MariaDB/Auror / HANA / Teradata / [✓ Snowflake] / Greenplum / Redshift / Hive ACID / File / FTP / Sharepoint / Azure DLS

Database Connection

Server	www.snowflakecomputing.com
Port	443
Role	admin
Warehouse	snowflakewarehouse
Database	mytestdb
Schema	snowflakeschema
User	hvruser
Password	●●●●●●●●●●●

Linux

Driver Manager Library

ODBCSYSINI

ODBC Driver

Field	Description
Database Connection	
Server	The hostname or ip-address of the machine on which the Snowflake server is running. **Example:** www.snowflakecomputing.com
Port	The port on which the Snowflake server is expecting connections. **Example:** 443
Role	The name of the Snowflake role to use. **Example:** admin
Warehouse	The name of the Snowflake warehouse to use. **Example:** snowflakewarehouse
Database	The name of the Snowflake database. **Example:** mytestdb
Schema	The name of the default Snowflake schema to use. **Example:** snowflakeschema
User	The username to connect HVR to the Snowflake **Database**. **Example:** hvruser
Password	The password of the **User** to connect HVR to the Snowflake **Database**.

Linux / Unix	
Driver Manager Library	The optional directory path where the ODBC Driver Manager Library is installed. This field is applicable only for Linux/Unix operating system. For a default installation, the ODBC Driver Manager Library is available at **/usr/lib64** and does not need to be specified. However, when UnixODBC is installed in for example **/opt/unixodbc** the value for this field would be **/opt/unixodbc/lib**.
ODBCSYSINI	The optional directory path where **odbc.ini** and **odbcinst.ini** files are located. This field is applicable only for Linux/Unix operating system. For a default installation, these files are available at **/etc** and do not need to be specified. However, when UnixODBC is installed in for example **/opt/unixodbc** the value for this field would be **/opt/unixodbc/etc**. The **odbcinst.ini** file should contain information about the Snowflake ODBC Driver under the heading **[SnowflakeDSIIDriver]**.
ODBC Driver	The user defined (installed) ODBC driver to connect HVR to the Snowflake server.

Integrate and Refresh Target

HVR supports integrating changes into Snowflake location. This section describes the configuration requirements for integrating changes (using **Integrate** and **refresh**) into Snowflake location. For the list of supported Snowflake versions, into which HVR can integrate changes, see Integrate changes into location in Capabilities.

HVR uses the Snowflake ODBC driver to write data to Snowflake during continuous **Integrate** and row-wise **Refresh**. However, the preferred methods for writing data to Snowflake are **Integrate** with **/Burst** and Bulk **Refresh** using staging as they provide better performance (see section 'Burst Integrate and Bulk Refresh' below).

When performing the **refresh** operation using slicing (option **-S**), a refresh job is created per each slice for refreshing only rows contained in the slice. These refresh jobs must not be run in parallel but should be scheduled one after another to avoid a risk of corruption on a Snowflake target location.

Grants for Integrate and Refresh Target

- The **User** should have permission to read and change replicated tables.

```
grant all privileges on future tables in database database to role hvrrole;
grant select, insert, update, delete, truncate on future tables in
database database to role hvrrole;
```

- The **User** should have permission to create and drop HVR state tables.
- The **User** should have permission to create and drop tables when HVR Refresh will be used to create target tables.

```
grant usage, modify, create table on schema schema in database database to role hvrrole
```

Burst Integrate and Bulk Refresh

While **Integrate** is running with parameter **/Burst** and Bulk **Refresh**, HVR can stream data into a target database straight over the network into a bulk loading interface specific for each DBMS, or else HVR puts data into a temporary directory (‘staging file') before loading data into a target database. For more information about staging files on Snowflake, see Snowflake Documentation.

For best performance, HVR performs **Integrate** with **/Burst** and Bulk **Refresh** into Snowflake using staging files. HVR implements **Integrate** with **/Burst** and Bulk **Refresh** (with file staging) into Snowflake as follows:

1. HVR first stages data into the configured staging platform:
    - Snowflake Internal Staging using Snowflake ODBC driver (**default**) **Since** v5.6.5/12
    - AWS or Google Cloud Storage using cURL library
    - Azure Blob FS using HDFS-compatible libhdfs API
2. HVR then uses Snowflake SQL command '**copy into**' to ingest data from the staging directories into the Snowflake target tables

HVR supports the following cloud platforms for staging files:

- Snowflake Internal Staging
- Snowflake on AWS
- Snowflake on Azure
- Snowflake on Google Cloud Storage

Snowflake Internal Staging

Since v5.6.5/12

By default, HVR stages data on the Snowflake internal staging before loading it into Snowflake while performing **Integrate** with **Burst** and **Bulk Refresh**. To use the Snowflake internal staging, it is not required to define action **LocationProperties** on the corresponding Integrate location.

Snowflake on AWS

HVR can be configured to stage the data on AWS S3 before loading it into Snowflake. For staging the data on AWS S3 and perform **Integrate** with **Burst** and **Bulk Refresh**, the following are required:

1. An AWS S3 location (bucket) - to store temporary data to be loaded into Snowflake. For more information about creating and configuring an S3 bucket, refer to AWS Documentation.
2. An AWS user with **AmazonS3FullAccess** permission policy - to access the S3 bucket. Alternatively, an AWS user with minimal set of permission can also be used.

    - **s3:GetBucketLocation**
    - **s3:ListBucket**
    - **s3:ListBucketMultipartUploads**
    - **s3:AbortMultipartUpload**
    - **s3:GetObject**
    - **s3:PutObject**
    - **s3:DeleteObject**

```
{
    "Statement": [
        {
            "Sid": <identifier>,
            "Effect": "Allow",
            "Principal": {
                    "AWS": "arn:aws:iam::<account_id>:<user>
/<username>",
            },
            "Action": [
                    "s3:GetObject",
                    "s3:GetObjectVersion",
                    "s3:PutObject",
                    "s3:DeleteObject",
                    "s3:DeleteObjectVersion",
                    "s3:AbortMultipartUpload"
            ],
            "Resource": "arn:aws:s3:::<bucket_name>/*"
        },
        {
            "Sid": <identifier>,
            "Effect": "Allow",
            "Principal": {
                    "AWS": "arn:aws:iam::<account_id>:<user>
/<username>"
            },
            "Action": [
                    "s3:ListBucket",
                    "s3:GetBucketLocation",
                    "s3:ListBucketMultipartUploads"
            ],
            "Resource": "arn:aws:s3:::<bucket_name>"
        }
    ]
}
```

For more information on the Amazon S3 permissions policy, refer to the AWS S3 documentation.

For more information, refer to the following AWS documentation:

- Amazon S3 and Tools for Windows PowerShell
- Managing Access Keys for IAM Users
- Creating a Role to Delegate Permissions to an AWS Service

3. Define action **LocationProperties** on the Snowflake location with the following parameters:
   - **/StagingDirectoryHvr**: the location where HVR will create the temporary staging files (ex. **s3://my_bucket_name/**).
   - **/StagingDirectoryDb**: the location from where Snowflake will access the temporary staging files. If **/StagingDirectoryHvr** is an Amazon S3 location then the value for **/StagingDirectoryDb** should be same as **/StagingDirectoryHvr**.
   - **/StagingDirectoryCredentials**: the AWS security credentials. For AWS, multiple formats are available for supplying the credentials (including encryption). For more information see **LocationProperties /StagingDirectoryCredentials**.
     Following are the two basic formats:
     - **role=***AWS_role*
     - **aws_access_key_id=***key***;aws_secret_access_key=***secret_key*
       E.g.: **aws_access_key_id=TWENDUIOWNDH1Q; aws_secret_access_key=N5D98Ae0EssvnPaBXlCJiBmpyMpmiD**

     For more information about getting your AWS credentials or Instance Profile Role, refer to AWS Documentation.

By default, HVR connects to **us-east-1** once for determining your S3 bucket region. If a firewall restriction or a service such as Amazon Private Link is preventing the determination of your S3 bucket region, you can change this region (**us-east-1**) to the region where your S3 bucket is located by defining the following action:

Group	Table	Action
Snowflake	*	**Environment/Name**=**HVR_S3_BOOTSTRAP_REGION /Value**= *s3_bucket_region*

Snowflake on Azure

Since v5.5.5/4

HVR can be configured to stage the data on Azure BLOB storage before loading it into Snowflake. For staging the data on Azure BLOB storage and perform **Integrate** with **Burst** and **Bulk Refresh**, the following are required:

1. An Azure BLOB storage location - to store temporary data to be loaded into Snowflake
2. An Azure user (storage account) - to access this location. For more information, refer to the Azure Blob storage documentation.
3. Define action **LocationProperties** on the Snowflake location with the following parameters:
    - **/StagingDirectoryHvr**: the location where HVR will create the temporary staging files (e.g. **wasbs://myblobcontainer**).
    - **/StagingDirectoryDb**: the location from where Snowflake will access the temporary staging files. If **/StagingDirectoryHvr** is an Azure location, this parameter should have the same value.
    - **/StagingDirectoryCredentials**: the Azure security credentials. The supported format for supplying the credentials is **azure_account=***azure_account***;** **azure_secret_access_key=***secret_key*.
      E.g.: **azure_account=mystorageaccount; azure_secret_access_key=t34RdYhph5bnIB7tWiKy5kDPteVJN4BTzBne3EDOP7KQBpuD**
4. For Linux (x64) and Windows (x64), since HVR 5.7.0/8 and 5.7.5/4, it is not required to install and configure the Hadoop client. However, if you want to use the Hadoop client, set the environment variable **HVR_AZURE_USE_HADOOP**=**1** and follow the steps mentioned below.

    For HVR versions prior to 5.7.0/8 or 5.7.5/4, the Hadoop Client must be present on the machine from which HVR will access the Azure Blob FS.
    Internally, HVR uses the WebHDFS REST API to connect to the Azure Blob FS. Azure Blob FS locations can only be accessed through HVR running on Linux or Windows, and it is not required to run HVR installed on the Hadoop NameNode although it is possible to do so. For more information about installing Hadoop client, refer to Apache Hadoop Releases.

    #### Hadoop Client Configuration

    The following are required on the machine from which HVR connects to Azure Blob FS:

    - Hadoop 2.6.x client libraries with Java 7 Runtime Environment or Hadoop 3.x client libraries with Java 8 Runtime Environment. For downloading Hadoop, refer to Apache Hadoop Releases.
    - Set the environment variable **$JAVA_HOME** to the Java installation directory. Ensure that this is the directory that has a bin folder, e.g. if the Java bin directory is d:\java\bin, **$JAVA_HOME** should point to d:\java.
    - Set the environment variable **$HADOOP_COMMON_HOME** or **$HADOOP_HOME** or **$HADOOP_PREFIX** to the Hadoop installation directory, or the **hadoop** command line client should be available in the path.
    - One of the following configuration is recommended,
        - Set **$HADOOP_CLASSPATH=$HADOOP_HOME/share/hadoop/tools/lib/***
        - Create a symbolic link for **$HADOOP_HOME/share/hadoop/tools/lib** in **$HADOOP_HOME/share/hadoop/common** or any other directory present in classpath.

Since the binary distribution available in Hadoop website lacks Windows-specific executables, a warning about unable to locate **winutils.exe** is displayed. This warning can be ignored for using Hadoop library for client operations to connect to a HDFS server using HVR. However, the performance on integrate location would be poor due to this warning, so it is recommended to use a Windows-specific Hadoop distribution to avoid this warning. For more information about this warning, refer to Hadoop Wiki and Hadoop issue HADOOP-10051.

Verifying Hadoop Client Installation

To verify the Hadoop client installation,

a. The **HADOOP_HOME/bin** directory in Hadoop installation location should contain the hadoop executables in it.
b. Execute the following commands to verify Hadoop client installation:

```
$JAVA_HOME/bin/java -version
$HADOOP_HOME/bin/hadoop version
$HADOOP_HOME/bin/hadoop classpath
```

c. If the Hadoop client installation is verified successfully then execute the following command to check the connectivity between HVR and Azure Blob FS:

   To execute this command successfully and avoid the error "ls: Password fs.adl.oauth2.client.id not found", few properties needs to be defined in the file **core-site.xml** available in the hadoop configuration folder (for e.g., **<path>/hadoop-2.8.3/etc/hadoop**). The properties to be defined differs based on the **Mechanism** (authentication mode). For more information, refer to section 'Configuring Credentials' in Hadoop Azure Blob FS Support documentation.

```
$HADOOP_HOME/bin/hadoop fs -ls wasbs://containername@accountname.
blob.core.windows.net/
```

Verifying Hadoop Client Compatibility with Azure Blob FS

To verify the compatibility of Hadoop client with Azure Blob FS, check if the following JAR files are available in the Hadoop client installation location (**$HADOOP_HOME/share/hadoop/tools/lib**):

```
hadoop-azure-<version>.jar
azure-storage-<version>.jar
```

Snowflake on Google Cloud Storage

Since v5.6.5/7

HVR can be configured to stage the data on Google Cloud Storage before loading it into Snowflake. For staging the data on Google Cloud Storage and perform **Integrate** with **Burst** and **Bulk** **Refresh**, the following are required:

1. A Google Cloud Storage location - to store temporary data to be loaded into Snowflake
2. A Google Cloud user (storage account) - to access this location.
3. Configure the storage integrations to allow Snowflake to read and write data into a Google Cloud Storage bucket. For more information, see Configuring an Integration for Google Cloud Storage in Snowflake documentation.
4. Define action **LocationProperties** on the Snowflake location with the following parameters:

   - **/StagingDirectoryHvr**: the location where HVR will create the temporary staging files (e.g. **gs://mygooglecloudstorage_bucketname**).

- **/StagingDirectoryDb**: the location from where Snowflake will access the temporary staging files. If **/StagingDirectoryHvr** is a Google cloud storage location, this parameter should have the same value.
- **/StagingDirectoryCredentials**: Google cloud storage credentials. The supported format for supplying the credentials is **"gs_access_key_id=***key***;gs_secret_access_key=***secret_key***;gs_storage_integration=***integration_name_for_google_cloud_storage***"**.
  E.g.: **gs_access_key_id=GOG88NACDFGFXERTE2RQ5C7; gs_secret_access_key=CbjneJGS0iEqb92BDBikEGDYlQoDKgOe6oDSG; gs_storage_integration=my_integration**

Compare and Refresh Source

- The **User** should have permission to read replicated tables.

```
grant select on tbl to hvruser
```