Requirements for HDFS

Contents

HDFS

Capture	Hub	Integrate
✓	✗	✓

This section describes the requirements, access privileges, and other features of HVR when using Hadoop Distributed File System (HDFS) for replication. HVR supports the WebHDFS API for reading and writing files from and to HDFS. For information about compatibility and supported versions of HDFS with HVR platforms, see Platform Compatibility Matrix.

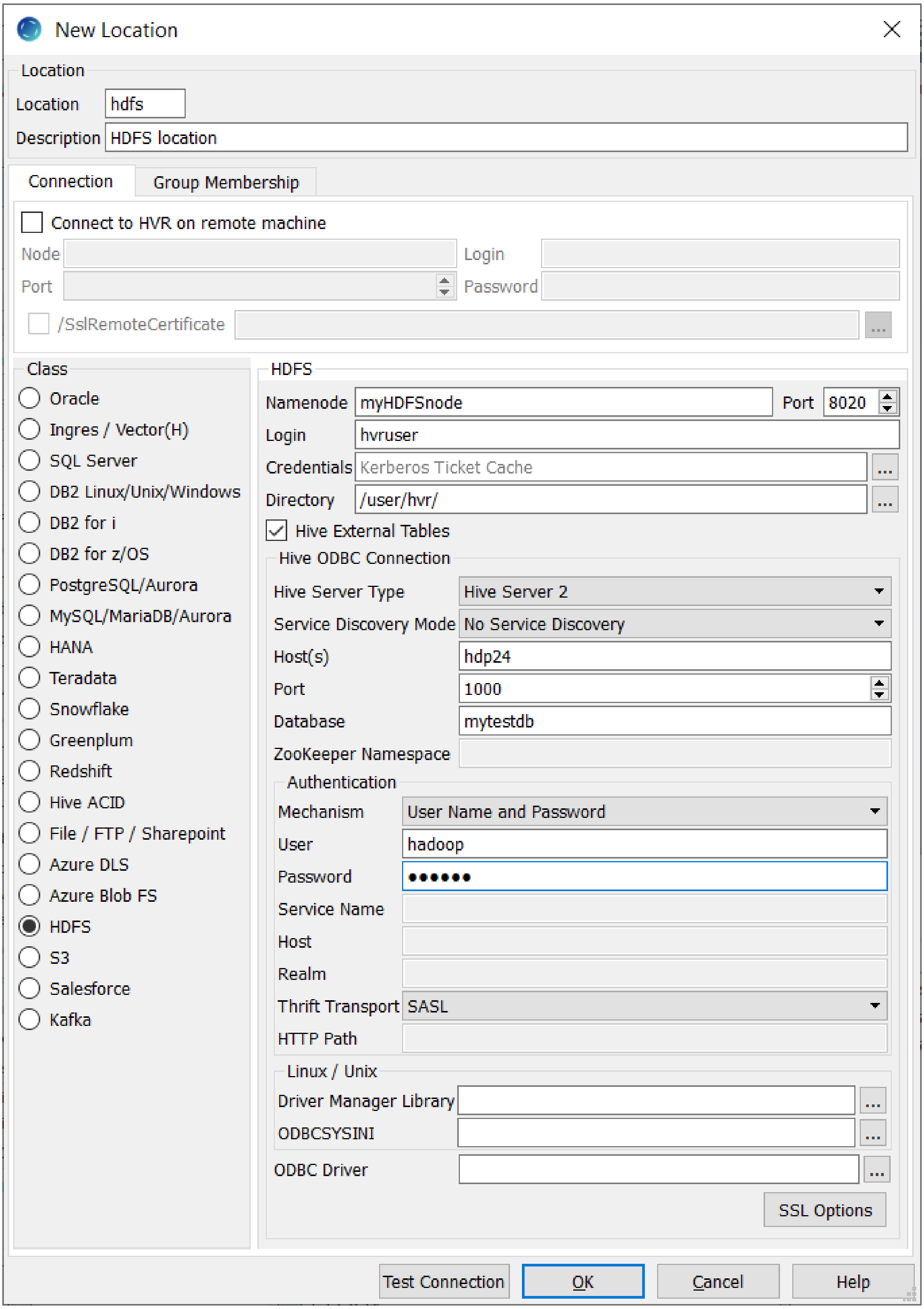
For the capabilities supported by HVR, see Capabilities.

For instructions to quickly setup replication into HDFS, see Quick Start for HVR - HDFS.

For requirements, access privileges, and other features of HVR when using MapR for replication, see Requirements for MapR.

Location Connection

This section lists and describes the connection details/parameters required for creating HDFS location in HVR.



New Location

Location

Location: hdfs

Description: HDFS location

Connection | Group Membership

☐ Connect to HVR on remote machine

Node | Login

Port | Password

☐ /SslRemoteCertificate

Class

○ Oracle
○ Ingres / Vector(H)
○ SQL Server
○ DB2 Linux/Unix/Windows
○ DB2 for i
○ DB2 for z/OS
○ PostgreSQL/Aurora
○ MySQL/MariaDB/Aurora
○ HANA
○ Teradata
○ Snowflake
○ Greenplum
○ Redshift
○ Hive ACID
○ File / FTP / Sharepoint
○ Azure DLS
○ Azure Blob FS
● HDFS
○ S3
○ Salesforce
○ Kafka

HDFS

Namenode: myHDFSnode

Port: 8020

Login: hvruser

Credentials: Kerberos Ticket Cache

Directory: /user/hvr/

☑ Hive External Tables

Hive ODBC Connection

Hive Server Type: Hive Server 2

Service Discovery Mode: No Service Discovery

Host(s): hdp24

Port: 1000

Database: mytestdb

ZooKeeper Namespace:

Authentication

Mechanism: User Name and Password

User: hadoop

Password: ●●●●●●

Service Name:

Host:

Realm:

Thrift Transport: SASL

HTTP Path:

Linux / Unix

Driver Manager Library:

ODBCSYSINI:

ODBC Driver:

Field	Description
Database Connection	
Namenode	The hostname of the HDFS NameNode. **Example:** myHDFSnode

Port	The port on which the HDFS server (**Namenode**) is expecting connections. **Example:** 8020
Login	The username to connect HVR to the HDFS **Namenode**. **Example:** hvruser
Credentials	The credential (Kerberos Ticket Cache file) for the username specified in **Login** to connect HVR to the HDFS **Namenode**. This field should be left blank to use a **keytab** file for authentication or if Kerberos is not used on the hadoop cluster. For more information about using Kerberos authentication, see HDFS Authentication and Kerberos.
Directory	The directory path in the HDFS **Namenode** to be used for replication. **Example:** /user/hvr/
Hive External Tables	Enable/Disable Hive ODBC connection configuration for creating Hive external tables above HDFS.

Hive ODBC Connection

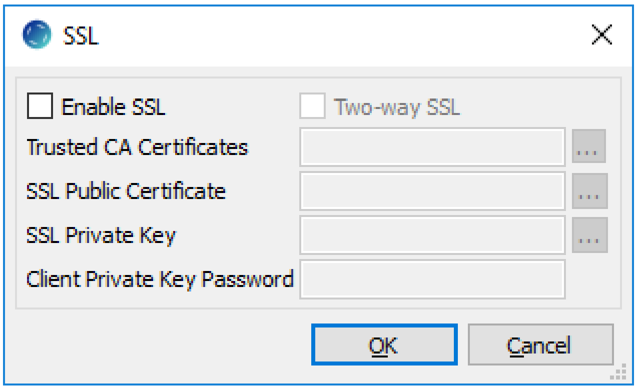
Following are the connection details/parameters required for connecting HVR to the Hive server.

Field	Description
Hive ODBC Connection	
Hive Server Type	The type of Hive server. Available options: • **Hive Server 1** (default): The driver connects to a Hive Server 1 instance. • **Hive Server 2**: The driver connects to a Hive Server 2 instance.
Service Discovery Mode	The mode for connecting to Hive. This field is enabled only if **Hive Server Type** is **Hive Server 2**. Available options: • **No Service Discovery** (default): The driver connects to Hive server without using the ZooKeeper service. • **ZooKeeper**: The driver discovers Hive Server 2 services using the ZooKeeper service.
Host(s)	The hostname or IP address of the Hive server. When Service Discovery Mode is ZooKeeper, specify the list of ZooKeeper servers in following format **[ZK_Host1]:[ZK_Port1],[ZK_Host2]:[ZK_Port2]**, where **[ZK_Host]** is the IP address or hostname of the ZooKeeper server and **[ZK _Port]** is the TCP port that the ZooKeeper server uses to listen for client connections. **Example:** hive-host
Port	The TCP port that the Hive server uses to listen for client connections. This field is enabled only if **Service Discovery Mode** is **No Service Discovery**. **Example:** 10000
Database	The name of the database schema to use when a schema is not explicitly specified in a query. **Example:** mytestdb
ZooKeeper Namespace	The namespace on ZooKeeper under which Hive Server 2 nodes are added. This field is enabled only if **Service Discovery Mode** is **ZooKeeper**.
Authentication	

Mechanism	The authentication mode for connecting HVR to **Hive Server 2**. This field is enabled only if **Hive Server Type** is **Hive Server 2**. Available options: • **No Authentication** (default) • **User Name** • **User Name and Password** • **Kerberos** • **Windows Azure HDInsight Service** Since v5.5.0/2
User	The username to connect HVR to Hive server. This field is enabled only if **Mechanism** is **User Name** or **User Name and Password**. **Example:** dbuser
Password	The password of the **User** to connect HVR to Hive server. This field is enabled only if **Mechanism** is **User Name and Password**.
Service Name	The Kerberos service principal name of the Hive server. This field is enabled only if **Mechanism** is **Kerberos**.
Host	The Fully Qualified Domain Name (FQDN) of the Hive Server 2 host. The value of **Host** can be set as **_HOST** to use the Hive server hostname as the domain name for Kerberos authentication. If **Service Discovery Mode** is disabled, then the driver uses the value specified in the Host connection attribute. If **Service Discovery Mode** is enabled, then the driver uses the **Hive Server 2** host name returned by ZooKeeper. This field is enabled only if **Mechanism** is **Kerberos**.
Realm	The realm of the Hive Server 2 host. It is not required to specify any value in this field if the realm of the Hive Server 2 host is defined as the default realm in Kerberos configuration. This field is enabled only if **Mechanism** is **Kerberos**.
Thrift Transport Since v5.5.0/2	The transport protocol to use in the Thrift layer. This field is enabled only if **Hive Server Type** is **Hive Server 2**. Available options: • **Binary** (This option is available only if **Mechanism** is **No Authentication** or **User Name and Password**.) • **SASL** (This option is available only if **Mechanism** is **User Name** or **User Name and Password** or **Kerberos**.) • **HTTP** (This option is not available if **Mechanism** is **User Name**.) For information about determining which Thrift transport protocols your Hive server supports, refer to HiveServer2 Overview and Setting Up HiveServer2 sections in Hive documentation.
HTTP Path Since v5.5.0/2	The partial URL corresponding to the Hive server. This field is enabled only if **Thrift Transport** is **HTTP**.
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**.
ODBC Driver	The user defined (installed) ODBC driver to connect HVR to the Hive server.
SSL Options	Show **SSL Options**.

SSL Options



Field	Description
Enable SSL	Enable/disable (one way) SSL. If enabled, HVR authenticates the Hive server by validating the SSL certificate shared by the Hive server.
Two-way SSL	Enable/disable two way SSL. If enabled, both HVR and Hive server authenticate each other by validating each others SSL certificate. This field is enabled only if **Enable SSL** is selected.
Trusted CA Certificates	The directory path where the **.pem** file containing the server's public SSL certificate signed by a trusted CA is located. This field is enabled only if **Enable SSL** is selected.
SSL Public Certificate	The directory path where the **.pem** file containing the client's SSL public certificate is located. This field is enabled only if **Two-way SSL** is selected.
SSL Private Key	The directory path where the **.pem** file containing the client's SSL private key is located. This field is enabled only if **Two-way SSL** is selected.
Client Private Key Password	The password of the private key file that is specified in **SSL Private Key**. This field is enabled only if **Two-way SSL** is selected.

Hadoop Client

HDFS 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. The Hadoop client should be present on the server from which HVR will access the HDFS. HVR uses HDFS compatible libhdfs API to connect, read and write data to HDFS during capture, integrate (continuous), refresh (bulk) and compare (direct file compare). For more information about installing Hadoop client, refer to Apache Hadoop Releases.

Hadoop Client Configuration

The following are required on the server from which HVR connects to HDFS:

- Install Hadoop 2.4.1 or later versions along with Java Runtime Environment:
  - Hadoop versions below 3.0 require JRE 7 or 8
  - Hadoop version 3.0 and higher requires only JRE 8

- 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.

  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,

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

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

3. If the Hadoop client installation is verified successfully then execute the following command to verify the connectivity between HVR and HDFS:

```
$HADOOP_HOME/bin/hadoop fs -ls hdfs://cluster/
```

Client Configuration Files

Client configuration files are required if Kerberos authentication is used in the Hadoop cluster or else they can be useful for debugging. Client configuration files contain settings for different services like HDFS, and others. If the HVR integrate server is not part of the cluster, it is recommended to download the configuration files for the cluster so that the Hadoop client knows how to connect to HDFS.

The client configuration files for Cloudera Manager or Ambari for Hortonworks can be downloaded from the respective cluster manager's web interface. For more information about downloading client configuration files, search for "Client Configuration Files" in the respective documentation for Cloudera and Hortonworks.

Hive External Tables

To **Compare** files that reside on the HDFS location, HVR allows you to create Hive external tables above HDFS. The connection details/parameters for Hive ODBC can be enabled for HDFS in the location creation screen by selecting the **Hive External Tables** field (see section Location Connection). For more information about configuring Hive external tables, refer to Apache Hadoop documentation.

ODBC Connection

HVR uses an ODBC connection to the Hadoop cluster for which it requires the ODBC driver (Amazon ODBC or HortonWorks ODBC) for Hive installed on the machine (or in the same network). The Amazon and HortonWorks ODBC drivers are similar and compatible to work with Hive 2.x release. However, it is recommended to use the Amazon ODBC driver for Amazon Hive and the Hortonworks ODBC driver for HortonWorks Hive. 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.

On Linux, HVR additionally requires unixODBC.

By default, HVR uses Amazon ODBC driver for connecting to Hadoop. To use the Hortonworks ODBC driver:

- For HVR versions since 5.3.1/25.1, use the **ODBC Driver** field available in the **New Location** screen to select the (user installed) Hortonworks ODBC driver.
- Prior to HVR 5.3.1/25.1, the following action definition is required:

Linux

Group	Table	Action
S3	*	**Environment /Name**=**HVR_ODBC_CONNECT_STRING_DRIVER /Value**=**Hortonworks Hive ODBC Driver 64-bit**

Windows

Group	Table	Action
S3	*	**Environment /Name**=**HVR_ODBC_CONNECT_STRING_DRIVER /Value**=**Hortonworks Hive ODBC Driver**

Channel Configuration

For the file formats (CSV, JSON, and AVRO) the following action definitions are required to handle certain limitations of the Hive deserialization implementation during Bulk or Row-wise **Compare**:

- For CSV

Group	Table	Action
S3	*	**FileFormat /NullRepresentation**=**\\N**
S3	*	**TableProperties /CharacterMapping**=**"\x00>\\0;\n>\\n;\r>\\r;">\""**
S3	*	**TableProperties /MapBinary**=**BASE64**

- For JSON

Group	Table	Action
S3	*	**TableProperties** /MapBinary=BASE64
S3	*	**FileFormat /JsonMode**=**ROW_FRAGMENTS**

- For Avro

Group	Table	Action
S3	*	**FileFormat /AvroVersion=v1_8**

v1_8 is the default value for **FileFormat /AvroVersion**, so it is not mandatory to define this action.

Integrate

HVR allows you to perform **HVR Refresh** or **Integrate** changes into an HDFS location. This section describes the configuration requirements for integrating changes (using **Integrate** or **HVR Refresh**) into the HDFS location.

Customize Integrate

Defining action **Integrate** is sufficient for integrating changes into an HDFS location. However, the default file format written into a target file location is HVR's own XML format and the changes captured from multiple tables are integrated as files into one directory. The integrated files are named using the integrate timestamp.

You may define other actions for customizing the default behavior of integration mentioned above. Following are few examples that can be used for customizing integration into the HDFS location:

Group	Table	Action	Annotation
HDFS	*	**FileFormat**	This action may be defined to: • specify the format (**Xml**, **Csv**, **Avro**, **Json**, or **Parquet**) of the files integrated into the target location. • escape any delimiters (e.g. comma) present in a column using the parameter **/QuoteCharacter**. • escape the quote character (**/QuoteCharacter**) defined, using the parameter **/EscapeCharacter**.
HDFS	*	**Integrate /RenameExpression**	To segregate and name the files integrated into the target location. For example, if **/RenameExpression={hvr_tbl_name}/{hvr_integ_tstamp}.csv** is defined, then for each table in the source, a separate folder (with the same name as the table name) is created in the target location, and the files replicated for each table are saved into these folders. This also enforces unique name for the files by naming them with a timestamp of the moment when the file was integrated into the target location.

HDFS	*	**ColumnProperties**	This action defines properties for a column being replicated. This action may be defined to: - integrate the delete operation. By default, for file-based target locations, HVR does not replicate the **delete** operation performed at the source location. So to integrate the delete operation, an extra column for timekey (**/TimeKey**) needs to be added in the target location. For this, action **ColumnProperties** may be defined with the following parameters: - **/Name**: This parameter defines the name for the extra column in the target location. - **/Extra**: This parameter defines that this is an extra column in the target location (a column which is not present in the source location). - **/IntegrateExpression**: This parameter defines the expression to be used for generating the **TimeKey** value. For example, **{hvr_integ_seq}** can be used here. This is a 36 byte string value (hex characters) which is unique and continuously increasing for a specific source location. - **/TimeKey**: This parameter defines that this is a **TimeKey** column. - **/Datatype=varchar**: This parameter defines the data type for the extra column. - **/Length=36**: This parameter defines the data type length for the extra column. - add the source operation type (using **hvr_op**) information in the target location. This action definition is required for performing **HVR Compare** if **ColumnProperties /TimeKey** column is defined on a target file location. For this, action **ColumnProperties** may be defined with the following parameters: - **/Name**: This parameter defines the name for the extra column in the target location. - **/Extra**: This parameter defines that this is an extra column in the target location (a column which is not present in the source location). - **/IntegrateExpression={hvr_op}**: This parameter defines the expression to be used for generating the information about source operation type. - **/Datatype=integer**: This parameter defines the data type for this extra column.

Integrate Limitations

By default, for file-based target locations, HVR does not replicate the **delete** operation performed at the source location.