



Denodo HDFS Custom Wrapper

Revision 20180918

NOTE

This document is confidential and proprietary of **Denodo Technologies**.
No part of this document may be reproduced in any form by any means without prior written authorization of **Denodo Technologies**.

Copyright © 2018
Denodo Technologies Proprietary and Confidential

CONTENTS

1 INTRODUCTION.....	5
2 HDFS.....	6
2.1 DELIMITED TEXT FILES.....	6
2.2 SEQUENCEFILES.....	6
2.3 MAPFILES.....	6
2.4 AVRO FILES.....	7
2.5 PARQUET FILES.....	7
3 USAGE.....	8
3.1 IMPORTING THE CUSTOM WRAPPER INTO VDP.....	8
3.2 CREATING AN HDFS DATA SOURCE.....	8
3.3 CREATING A BASE VIEW.....	9
4 AMAZON S3.....	29
4.1 CONFIGURING S3 AUTHENTICATION PROPERTIES.....	29
4.2 CONFIGURING S3N AUTHENTICATION PROPERTIES.....	29
4.3 CONFIGURING S3A AUTHENTICATION PROPERTIES.....	30
4.4 SIGNATURE VERSION 4 SUPPORT.....	31
5 AZURE DATA LAKE STORE.....	32
5.1 CONFIGURING AUTHENTICATION PROPERTIES.....	32
6 AZURE BLOB STORAGE.....	33
6.1 CONFIGURING AUTHENTICATION PROPERTIES.....	33
7 HDFS COMPRESSED FILES.....	34
8 SECURE CLUSTER WITH KERBEROS.....	35
9 TROUBLESHOOTING.....	38
10 APPENDICES.....	41
10.1 HOW TO USE THE HADOOP VENDOR'S CLIENT LIBRARIES.....	41

1 INTRODUCTION

The HDFS Custom Wrapper distribution contains five Virtual DataPort custom wrappers capable of reading several file formats stored in **Hadoop Distributed File System** (HDFS).

Supported formats are:

- Delimited text files
- Sequence files
- Map files
- Avro files
- Parquet files

2 HDFS

The Hadoop Distributed File System is a distributed, scalable, and portable file system used by the Hadoop platform.

In HDFS, data is divided into blocks and copies of these blocks are stored on other servers in the Hadoop cluster. That is, an individual file is actually stored as smaller blocks that are replicated across multiple servers in the entire cluster. This redundancy offers multiple benefits for Big Data processing:

- Higher availability.
- Better scalability: map and reduce functions can be executed on smaller subsets of large data sets.
- Data locality: move the computation closer to the data to reduce latency.

2.1 DELIMITED TEXT FILES

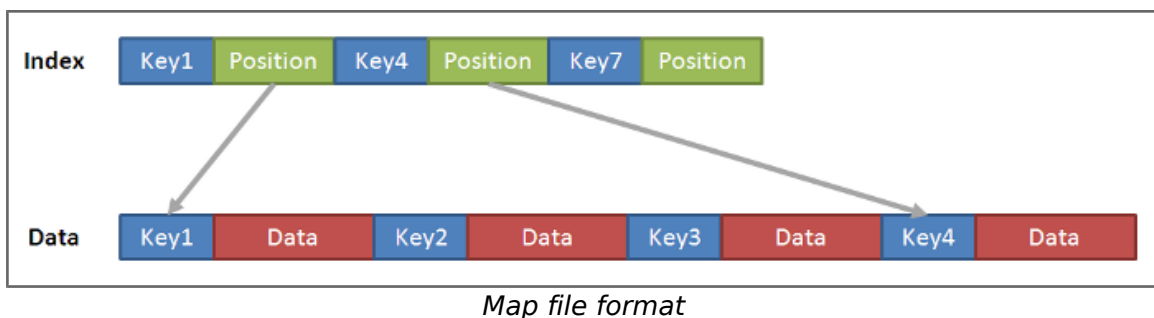
Delimited text files store plain text and each line has values separated by a delimiter, such as tab, space, comma, etc.

2.2 SEQUENCEFILES

Sequence files are binary record-oriented files, where each record has a serialized key and a serialized value.

2.3 MAPFILES

A map is a directory containing two sequence files. The data file (/data) is identical to the sequence file and contains the data stored as binary key/value pairs. The index file (/index), which contains a key/value map with seek positions inside the data file to quickly access the data.



2.4 AVRO FILES

Avro data files are self-describing, containing the full schema for the data in the file. An Avro schema is defined using JSON. The schema allows you to define two types of data:

- primitive data types: string, integer, long, float, double, byte, null and boolean.
- complex type definitions: a record, an array, an enum, a map, a union or a fixed type.

```
{
  "namespace": "example.avro",
  "type": "record",
  "name": "User",
  "fields": [
    { "name": "name", "type": "string" },
    { "name": "favorite_number", "type": [ "int", "null" ] },
    { "name": "favorite_color", "type": [ "string", "null" ] }
  ]
}
```

Avro schema

2.5 PARQUET FILES

Parquet is a column-oriented data store of the Hadoop ecosystem. It provides data compression on a per-column level and encoding schemas.

The data are described by a schema that starts with the word `Message` and contains a group of fields. Each field is defined by a *repetition* (required, optional, or repeated), a *type* and a *name*.

```
Message Customer {
  required int32 id;
  required binary firstname (UTF8);
  required binary lastname (UTF8);
}
```

Parquet schema

Primitives types in parquet are boolean, int32, int64, int96, float, double, binary and fixed_len_byte_array. There are no String types but there are logical types which allows interpreting binaries as a String, JSON or other types.

Complex types are defined by a group type, which adds a layer of nesting.

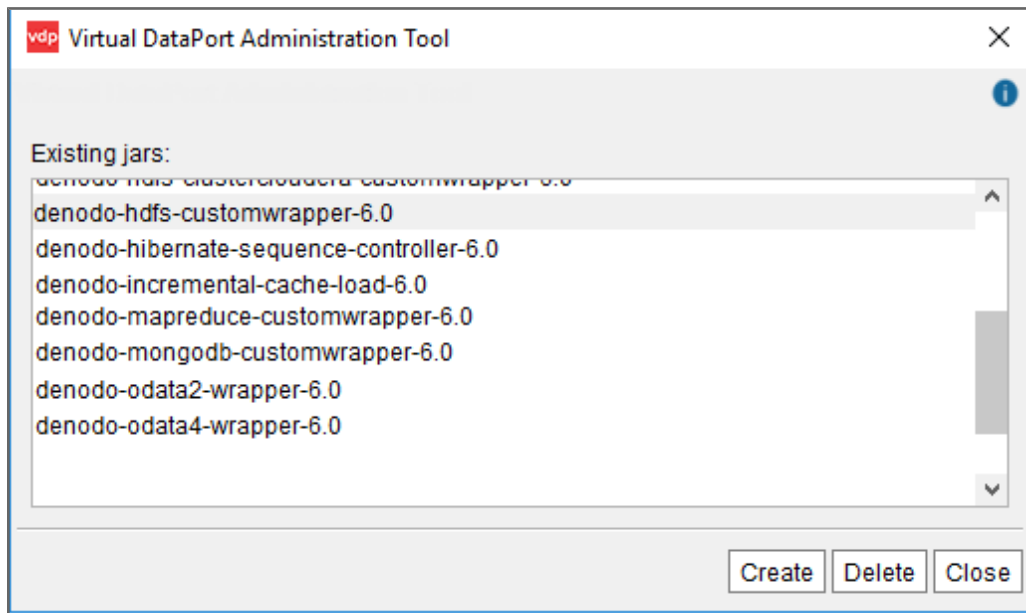
3 USAGE

3.1 IMPORTING THE CUSTOM WRAPPER INTO VDP

In order to use the HDFS Custom Wrapper in VDP, we must configure the Admin Tool to import the extension.

From the HDFS Custom Wrapper distribution, we will select the `denodo-hdfs-customwrapper-${version}-jar-with-dependencies.jar` file and upload it to VDP.

No other jars are required as this one will already contain all the required dependencies.



HDFS Extension in VDP

3.2 CREATING AN HDFS DATA SOURCE

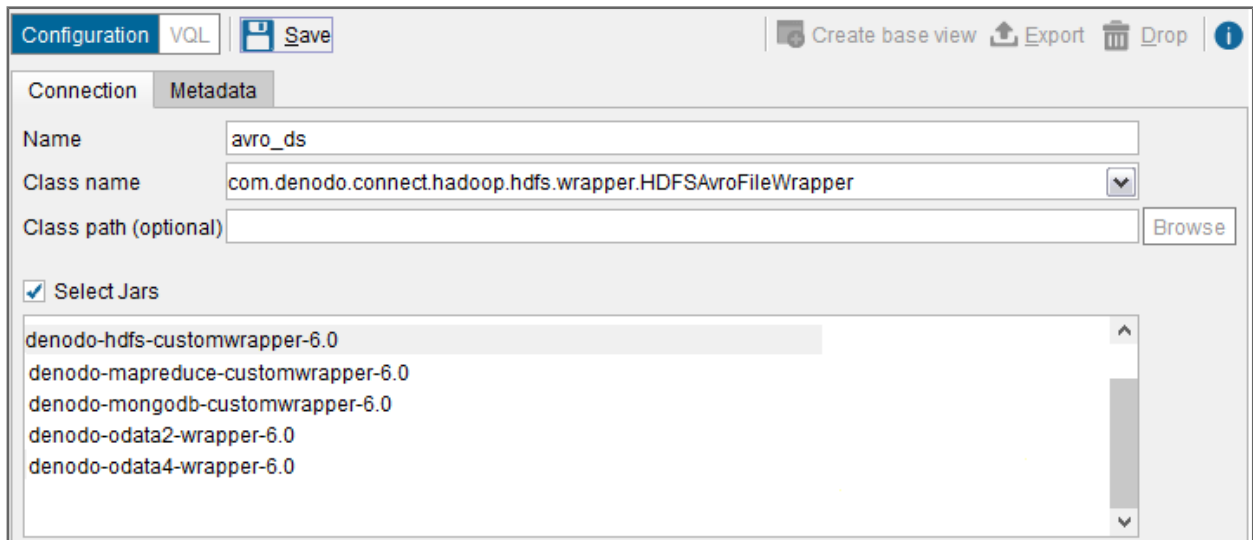
Once the custom wrapper jar file has been uploaded to VDP using the Admin Tool, we can create new data sources for this custom wrapper --and their corresponding base views-- as usual.

Go to New → Data Source → Custom and specify one of the possible wrappers:

- `com.denodo.connect.hadoop.hdfs.wrapper.HDFSDelimitedTextFileWrapper`
- `com.denodo.connect.hadoop.hdfs.wrapper.HDFSSequenceFileWrapper`
- `com.denodo.connect.hadoop.hdfs.wrapper.HDFSMapFileWrapper`

- `com.denodo.connect.hadoop.hdfs.wrapper.HDFSAvroFileWrapper`
- `com.denodo.connect.hadoop.hdfs.wrapper.WebHDFSFileWrapper`
- `com.denodo.connect.hadoop.hdfs.wrapper.HDFSParquetFileWrapper`

Also check 'Select Jars' and select the jar file of the custom wrapper.



HDFS Data Source

3.3 CREATING A BASE VIEW

Once the custom wrapper has been registered, we will be asked by VDP to create a base view for it.

3.3.1 HDFSDelimitedTextFileWrapper

Custom wrapper for reading delimited text files stored in HDFS. Its base views need the following parameters:

- **File system URI:** A URI whose scheme and authority identify the file system.
 - **HDFS:** `hdfs://<ip>:<port>`.
 - **Amazon S3:** `s3a://@<bucket>`. For configuring the credentials see **Amazon S3** section.
 - **Azure Data Lake Store:** `adl://<account name>.azuredatalakestore.net/`. For configuring the credentials see **Azure Data Lake Store** section.
 - **Azure Blob Storage:** `wasb://<container>@<account>.blob.core.windows.net`. For configuring the credentials see **Azure Blob Storage** section.

! Note

If you enter a literal that contains one of the special characters used to indicate interpolation variables @, \, ^, {, }, you have to escape these characters with \.

E.g if the URI contains @ you have to enter \@.

- **Path:** input path for the delimited file or the directory containing the files.
- **File name pattern:** If you want this wrapper to only obtain data from some of the files of the directory, you can enter a regular expression that matches the names of these files.
For example, if you want the base view to return the data of all the files with the extension csv set the File name pattern to (.*)\.csv. Optional.
- **Delete after reading:** Requests that the file or directory denoted by the path be deleted when the wrapper terminates.
- **Custom core-site.xml file:** configuration file that overrides the default core parameters. Optional.
- **Custom hdfs-site xml file:** configuration file that overrides the default HDFS parameters. Optional.
- **Separator:** delimiter between the values of a row. Default is the comma (,) and cannot be a line break (\n or \r). Optional.

Some “invisible” characters have to be entered in a special way:

Character	Meaning
\t	Tab
\f	Formfeed

! Note

If you enter a literal that contains one of the special characters used to indicate interpolation variables @, \, ^, {, }, you have to escape these characters with \.

E.g if the separator is the tab character \t you have to enter \\t.

- **Quote:** Character used to encapsulate values containing special characters. Default is quote ("). Optional.
- **Comment marker:** Character marking the start of a line comment. Comments are

disabled by default. Optional.

- **Escape:** Escape character. Escapes are disabled by default. Optional.
- **Ignore spaces:** Whether spaces around values are ignored. False by default.
- **Header:** If selected, the wrapper considers that the first line contains the names of the fields in this file. These names will be the fields' names of the base views created from this wrapper. True by default.

File system URI	<input type="text" value="hdfs://quickstart.cloudera:8020"/>	
Path	<input type="text" value="/user/cloudera/df/SearchLog.tsv"/>	
File name pattern	<input type="text" value="(.*)\\.tsv"/>	
	<input type="checkbox"/> Delete after reading	
Custom core-site.xml file	<input type="text" value="None"/>	<input type="button" value="Configure"/>
Custom hdfs-site.xml file	<input type="text" value="None"/>	<input type="button" value="Configure"/>
Separator	<input type="text" value="\\t"/>	
Quote	<input type="text"/>	
Comment marker	<input type="text"/>	
Escape	<input type="text"/>	
	<input type="checkbox"/> Ignore spaces	
	<input type="checkbox"/> Header	
	<input type="checkbox"/> Kerberos enabled	
Kerberos principal name	<input type="text"/>	
Kerberos keytab file	<input type="text" value="None"/>	<input type="button" value="Configure"/>
Kerberos password	<input type="text"/>	
Kerberos Distribution Center	<input type="text"/>	

HDFSDeimitedTextFileWrapper base view edition

View schema:	Field Name	Field Type
	playerid	text
	yearid	text
	stint	text
	teamid	text
	lgid	text
	pos	text
	g	text
	gs_0	text
	innouts	text
	po	text
	a	text
	e	text
	dp	text
	pb	text
	wp	text
	sb	text
	cs	text
	zr	text

View schema

The execution of the wrapper returns the values contained in the file or group of files, if the Path input parameter denotes a directory.

Total rows received: 167938 (shown 150)							
playerid	yearid	stint	teamid	lgid	pos	g	g
abercda01	1871	1	TRO	NA	SS	1	
addybo01	1871	1	RC1	NA	2B	22	
addybo01	1871	1	RC1	NA	SS	3	
allisar01	1871	1	CL1	NA	2B	2	
allisar01	1871	1	CL1	NA	OF	29	
allisdo01	1871	1	WS3	NA	C	27	
ansonca01	1871	1	RC1	NA	1B	1	
ansonca01	1871	1	RC1	NA	2B	2	
ansonca01	1871	1	RC1	NA	3B	20	
ansonca01	1871	1	RC1	NA	C	5	
ansonca01	1871	1	RC1	NA	OF	1	

View results

3.3.2 HDFSSequenceFileWrapper

Custom wrapper for reading sequence files stored in HDFS. Its base views need the following parameters:

- **File system URI:** A URI whose scheme and authority identify the file system.
 - **HDFS:** hdfs://<ip>:<port>.
 - **Amazon S3:** s3a://@<bucket>. For configuring the credentials see **Amazon S3** section.
 - **Azure Data Lake Store:**
adl://<account name>.azuredatalakestore.net/
For configuring the credentials see **Azure Data Lake Store** section.
 - **Azure Blob Storage:**
wasb://<container>@<account>.blob.core.windows.net
For configuring the credentials see **Azure Blob Storage** section.

! Note

If you enter a literal that contains one of the special characters used to indicate interpolation variables @, \, ^, {, }, you have to escape these characters with \.

E.g if the URI contains @ you have to enter \@.

- **Path:** input path for the sequence file or the directory containing the files.
- **File name pattern:** If you want this wrapper to only obtain data from some of the files of the directory, you can enter a regular expression that matches the names of these files.
For example, if you want the base view to return the data of all the files with the extension seq set the File name pattern to (.*)\.seq. Optional.
- **Delete after reading:** Requests that the file or directory denoted by the path be deleted when the wrapper terminates.
- **Custom core-site.xml file:** configuration file that overrides the default core parameters. Optional.
- **Custom hdfs-site.xml file:** configuration file that overrides the default HDFS parameters. Optional.
- **Key class:** key class name implementing org.apache.hadoop.io.Writable interface.
- **Value class:** value class name implementing org.apache.hadoop.io.Writable interface.

File system URI	<input type="text" value="h3n://AKIAJXVIEYD2Q74CGMTA:SALdz9RIETUxCOlvd4eVOkgGT5FnkmGvt1"/>		
Path	<input type="text" value="/sequencefil/sequence.seq"/>		
File name pattern	<input type="text"/>		
	<input type="checkbox"/> Delete after reading		
Custom core-site.xml file	<input type="text" value="None"/>	<input type="button" value="Configure"/>	
Custom hdfs-site.xml file	<input type="text" value="None"/>	<input type="button" value="Configure"/>	
Key class	<input type="text" value="org.apache.hadoop.io.IntWritable"/>		
Value class	<input type="text" value="org.apache.hadoop.io.Text"/>		
	<input type="checkbox"/> Kerberos enabled		
Kerberos principal name	<input type="text"/>		
Kerberos keytab file	<input type="text" value="None"/>	<input type="button" value="Configure"/>	
Kerberos password	<input type="text"/>		
Kerberos Distribution Center	<input type="text"/>		

HDFSSequenceFileWrapper base view edition

View schema:	Field Name	Field Type
	key	int
	value	text

View schema

The execution of the wrapper returns the key/value pairs contained in the file or group of files, if the Path input parameter denotes a directory.

Total rows received: 100 (shown 100)	
key	value
100	One, two, buckle my shoe
99	Three, four, shut the door
98	Five, six, pick up sticks
97	Seven, eight, lay them straight
96	Nine, ten, a big fat hen
95	One, two, buckle my shoe
94	Three, four, shut the door
93	Five, six, pick up sticks
92	Seven, eight, lay them straight
91	Nine, ten, a big fat hen
90	One, two, buckle my shoe
89	Three, four, shut the door

View results

3.3.3 HDFSMapFileWrapper

Custom wrapper for reading map files stored in HDFS. Its base views need the following parameters:

- **File system URI:** A URI whose scheme and authority identify the file system.
 - **HDFS:** `hdfs://<ip>:<port>`.
 - **Amazon S3:** `s3a://<bucket>`. For configuring the credentials see **Amazon S3** section.
 - **Azure Data Lake Store:**
`adl://<account name>.azuredatalakestore.net/`
For configuring the credentials see **Azure Data Lake Store** section.
 - **Azure Blob Storage:**
`wasb://<container>@<account>.blob.core.windows.net`
For configuring the credentials see **Azure Blob Storage** section.

! Note

If you enter a literal that contains one of the special characters used to indicate interpolation variables `@`, `\`, `^`, `{`, `}`, you have to escape these characters with `\`.

E.g if the URI contains `@` you have to enter `\@`.

- **Path:** input path for the directory containing the map file. Also the path to the index or data file could be specified. When using **Amazon S3**, a flat file system where there is no folder concept, **the path to the index or data should be used**.
- **File name pattern:** If you want this wrapper to only obtain data from some of the files of the directory, you can enter a regular expression that matches the names of these files.
For example, if you want the base view to return the data of all the files with the extension whatever set the File name pattern to `(.*)\.`whatever. Optional.
- **Delete after reading:** Requests that the file or directory denoted by the path be deleted when the wrapper terminates.
- **Custom core-site.xml file:** configuration file that overrides the default core parameters. Optional.
- **Custom hdfs-site.xml file:** configuration file that overrides the default HDFS parameters. Optional.
- **Key class:** key class name implementing the `org.apache.hadoop.io.WritableComparable` interface. `WritableComparable` is used because records are sorted in **key order**.
- **Value class:** value class name implementing the `org.apache.hadoop.io.Writable` interface.

File system URI	<input type="text" value="hdfs://quickstart.cloudera:8020"/>	
Path	<input type="text" value="/user/cloudera/MAP"/>	
File name pattern	<input type="text"/>	
	<input type="checkbox"/> Delete after reading	
Custom core-site.xml file	<input type="text" value="None"/>	<input type="button" value="Configure"/>
Custom hdfs-site.xml file	<input type="text" value="None"/>	<input type="button" value="Configure"/>
Key class	<input type="text" value="org.apache.hadoop.io.IntWritable"/>	
Value class	<input type="text" value="org.apache.hadoop.io.Text"/>	
	<input type="checkbox"/> Kerberos enabled	
Kerberos principal name	<input type="text"/>	
Kerberos keytab file	<input type="text" value="None"/>	<input type="button" value="Configure"/>
Kerberos password	<input type="text"/>	
Kerberos Distribution Center	<input type="text"/>	

HDFSMapFileWrapper base view edition

View schema:	Field Name	Field Type
	key	int
	value	text

View schema

The execution of the wrapper returns the key/value pairs contained in the file or group of files, if the Path input parameter denotes a directory.

Total rows received: 1024 (shown 150)	
△ ▾	
key	value
1	One, two, buckle my shoe
2	Three, four, shut the door
3	Five, six, pick up sticks
4	Seven, eight, lay them straight
5	Nine, ten, a big fat hen
6	One, two, buckle my shoe
7	Three, four, shut the door
8	Five, six, pick up sticks
9	Seven, eight, lay them straight
10	Nine, ten, a big fat hen
11	One, two, buckle my shoe
12	Three, four, shut the door

View results

3.3.4 HDFSAvroFileWrapper

Custom wrapper for reading Avro files stored in HDFS. Its base views need the following parameters:

- **File system URI:** A URI whose scheme and authority identify the file system.
 - **HDFS:** `hdfs://<ip>:<port>`.
 - **Amazon S3:** `s3a://@<bucket>`. For configuring the credentials see **Amazon S3** section.
 - **Azure Data Lake Store:**
`adl://<account name>.azuredatalakestore.net/`
For configuring the credentials see **Azure Data Lake Store** section.
 - **Azure Blob Storage:**
`wasb://<container>@<account>.blob.core.windows.net`
For configuring the credentials see **Azure Blob Storage** section.

! Note

If you enter a literal that contains one of the special characters used to indicate interpolation variables `@`, `\`, `^`, `{`, `}`, you have to escape these characters with `\`.

E.g if the URI contains `@` you have to enter `\@`.

- **File name pattern:** If you want this wrapper to only obtain data from some of the files of the directory, you can enter a regular expression that matches the names of these files.

For example, if you want the base view to return the data of all the files with the extension avro set the File name pattern to `(.*)\.avro`. Optional.

- **Delete after reading:** Requests that the file denoted by the path be deleted when the wrapper terminates.
- **Custom core-site.xml file:** configuration file that overrides the default core parameters. Optional.
- **Custom hdfs-site xml file:** configuration file that overrides the default HDFS parameters. Optional.

There is also two parameters that are **mutually exclusive**:

- **Avro schema path:** input path for the Avro schema file or
- **Avro schema JSON:** JSON of the Avro schema.

! Note

If you enter a literal that contains one of the special characters used to indicate interpolation variables `@`, `\`, `^`, `{`, `}` in the **Avro schema JSON** parameter, you have to escape these characters with `\`. For example:

```
\{
  "type": "map",
  "values": \{
    "type": "record",
    "name": "ATM",
    "fields": [
      \{ "name": "serial_no", "type": "string" \},
      \{ "name": "location", "type": "string" \}
    ]
  }
\}
```


File system URI	<input type="text" value="hdfs://quickstart.cloudera:8020"/>	
Avro schema path	<input type="text" value="/user/cloudera/avro/RecordWithAllTypes.avsc"/>	
Avro schema JSON	<input type="text"/>	
File name pattern	<input type="text" value="(.*).avro"/>	
	<input type="checkbox"/> Delete after reading	
Custom core-site.xml file	<input type="text" value="None"/>	<input type="button" value="Configure"/>
Custom hdfs-site.xml file	<input type="text" value="None"/>	<input type="button" value="Configure"/>
	<input type="checkbox"/> Kerberos enabled	
Kerberos principal name	<input type="text"/>	
Kerberos keytab file	<input type="text" value="None"/>	<input type="button" value="Configure"/>
Kerberos password	<input type="text"/>	
Kerberos Distribution Center	<input type="text"/>	

HDFSAvroFileWrapper base view edition

```
{
  "type" : "record",
  "name" : "Doc",
  "doc" : "adoc",
  "fields" : [ {
    "name" : "id",
    "type" : "string"
  }, {
    "name" : "user_friends_count",
    "type" : [ "int", "null" ]
  }, {
    "name" : "user_location",
    "type" : [ "string", "null" ]
  }, {
    "name" : "user_description",
    "type" : [ "string", "null" ]
  }, {
    "name" : "user_statuses_count",
    "type" : [ "int", "null" ]
  }, {
    "name" : "user_followers_count",
    "type" : [ "int", "null" ]
  }, {
    "name" : "user_name",
    "type" : [ "string", "null" ]
  }, {
    "name" : "user_screen_name",
    "type" : [ "string", "null" ]
  }, {
    "name" : "created_at",
    "type" : [ "string", "null" ]
  }, {
    "name" : "text",
    "type" : [ "string", "null" ]
  }, {
    "name" : "retweet_count",
    "type" : [ "int", "null" ]
  }, {
    "name" : "retweeted",
    "type" : [ "boolean", "null" ]
  }, {
    "name" : "in_reply_to_user_id",
    "type" : [ "long", "null" ]
  }, {
    "name" : "source",
    "type" : [ "string", "null" ]
  }, {
    "name" : "in_reply_to_status_id",
    "type" : [ "long", "null" ]
  }, {
    "name" : "media_url_https",
    "type" : [ "string", "null" ]
  }, {
    "name" : "expanded_url",
```

```
"type" : [ "string", "null" ]
} ] }
```

Content of the `/user/cloudera/schema.avsc` file

View schema:	Field Name	Field Type
	avrofilepath	text
	doc	avro_ds_doc
	id	text
	user_friends_count	int
	user_location	text
	user_description	text
	user_statuses_count	int
	user_followers_count	int
	user_name	text
	user_screen_name	text
	created_at	text
	text	text
	retweet_count	int
	retweeted	boolean
	in_reply_to_user_id	long
	source	text
	in_reply_to_status_id	long
	media_url_https	text
	expanded_url	text

View schema

The execution of the view returns the values contained in the Avro file specified in the WHERE clause of the VQL sentence:

```
SELECT * FROM avro_ds_file
WHERE avrofilepath = '/user/cloudera/file.avro'
```

Total rows received: 2104 (shown 150)	
<div> <div> <div></div> <div></div> </div> </div>	
avrofilepath	doc
/user/cloudera/file.avro	[Register]...
/user/cloudera/file.avro	[Register]...
/user/cloudera/file.avro	[Register]

RESU... → doc										
id	user_frien...	user_loca...	user_des...	user_stat...	user_follo...	user_name	user_scre...	created_at	text	re
10000	10000	location1...		1	1	fake user...	fake_user...	1985-05-...	tweet text ...	0

View results

After applying a flattening operation results are as follows.

Total rows received: 2104 (shown 150)

△ ▽

avrofilepath	id	user_frien...	user_locati...	user_desc...	user_statu...	user_follo...	user_name	user_scre...	cre
/user/cloud...	10000	10000	location10...		1	1	fake user1...	fake_user1...	198
/user/cloud...	10001	10001	location10...		1	1	fake user1...	fake_user1...	198

Flattened results

3.3.4.1 Field Projection

The recommended way for dealing with **projections** in HDFSAvroFileWrapper is by means of the JSON schema parameters:

- Avro schema path or
- Avro schema JSON

By giving to the wrapper a JSON schema containing exclusively the fields we are interested in, the reader used by the HDFSAvroFileWrapper will return to VDP only these fields, making the select operation faster.

If we configure the parameter Avro schema JSON with only some of the fields of the /user/cloudera/schema.avsc file used in the previous example, like in the example below (notice the escaped characters):

```
\{
  "type" : "record",
  "name" : "Doc",
  "doc" : "adoc",
  "fields" : [ \{
    "name" : "id",
    "type" : "string"
  }, \{
```

```

    "name" : "user_friends_count",
    "type" : [ "int", "null" ]
  }, \{
    "name" : "user_location",
    "type" : [ "string", "null" ]
  }, \{
    "name" : "user_followers_count",
    "type" : [ "int", "null" ]
  }, \{
    "name" : "user_name",
    "type" : [ "string", "null" ]
  }, \{
    "name" : "created_at",
    "type" : [ "string", "null" ]
  }
]
\}

```

Schema with the selected fields

the base view in VDP will contain a subset of the previous base view of the example: the ones matching the new JSON schema provided to the wrapper.

View schema:	Field Name	Field Type
	avroFilepath	text
Doc	avro_ds_Doc	
	id	text
	user_friends_count	int
	user_location	text
	user_followers_count	int
	user_name	text
	created_at	text

Base view with the selected fields

id	user_friends_count	user_location	user_followers_count	user_name	created_at
10000	10000	location10000	1	fake user10000	1985-05-11T10:09:19Z

View results with the selected fields

3.3.5 WebHDFSFileWrapper

Important

WebHDFSFileWrapper is **deprecated**.

- For XML, JSON and Delimited files the best alternative is using the **VDP standard data sources**, using the HTTP Client in its Data route parameter. These data sources offers a better solution for HTTP/HTTPs access as they include proxy access, SPNEGO authentication, OAuth2 etc.
- For Avro, Sequence, Map and Parquet files the best alternative is using the specific HDFS Custom Wrapper type: **HDFSAvroFileWrapper**, **HDFSSequenceFileWrapper**, **HDFSMapFileWrapper** or **HDFSParquetFileWrapper** with webhdfs scheme in their File system URI parameter. And placing their credentials as HDFS configuration options in the xml configuration files.

Custom wrapper for reading delimited text files stored in HDFS using the **WebHDFS**.

3.3.5.1 About WebHDFS

WebHDFS provides HTTP REST access to HDFS. It supports all HDFS user operations including reading files, writing to files, making directories, changing permissions and renaming.

The advantage of WebHDFS are:

- **Version-independent** REST-based protocol which means that can be read and written to/from Hadoop clusters no matter their version. This addresses the issue of using the Java API (RPC-based) that requires both the client and the Hadoop cluster to share the same version. Upgrading one without the other causes serialization errors meaning the client cannot interact with the cluster.
- Read and write data in HDFS in a cluster behind a firewall. A proxy WebHDFS (for example: HttpFS) could be use, it acts as a gateway and is the only system that is allowed to send and receive data through the firewall. The only difference between using or not the proxy will be in the host:port pair where the HTTP requests are issued:
 - Default port for WebHDFS is 50075.
 - Default port for HttpFS is 14000.

3.3.5.2 Custom wrapper

The base views created from the WebHDFSFileWrapper need the following parameters:

- **Host IP:** IP or <bucket>.s3.amazonaws.com for Amazon S3.
- **Host port:** HTTP port. Default port for WebHDFS is 50075. For HttpFS is 14000. For Amazon S3 is 80.

- **User:** The name of the the authenticated user when security is off. If is not set, the server may either set the authenticated user to a default web user, if there is any, or return an error response.
When using **Amazon S3** <id>:<secret> should be indicated.
- **Path:** input path for the delimited file.
- **Separator:** delimiter between values. Default is the comma.
- **Quote:** Character used to encapsulate values containing special characters. Default is quote.
- **Comment marker:** Character marking the start of a line comment. Comments are disable by default.
- **Escape:** Escape character. Escapes are disabled by default.
- **Ignore spaces:** Whether spaces around values are ignored. False by default.
- **Header:** Whether the file has a header or not. True by default.
- **Delete after reading:** Requests that the file or directory denoted by the path be deleted when the wrapper terminates.

Host IP	melkus.denodo.com
Host port	50070
User	
Path	/user/cloudera/csv/Master.csv
Separator	,
Quote	
Comment marker	
Escape	
	<input type="checkbox"/> Ignore spaces <input checked="" type="checkbox"/> Header <input type="checkbox"/> Delete after reading

WebHDFSFileWrapper base view edition

View schema:	Field Name	Field Type
	playerid	text
	birthyear	text
	birthmonth	text
	birthday	text
	birthcountry	text
	birthstate	text
	birthcity	text
	deathyear	text
	deathmonth	text
	deathday	text
	deathcountry	text
	deathstate	text
	deathcity	text
	namefirst	text
	namelast	text
	namegiven	text
	weight	text
	height	text
	bats	text
	throws	text

View schema

The execution of the wrapper returns the values contained in the file.

Total rows received: 18589 (shown 150)										
playerid	birthyear	birthmonth	birthday	birthcountry	birthstate	birthcity	deathyear	deathmonth	deathday	de
aardsda01	1981	12	27	USA	CO	Denver				
aaronha01	1934	2	5	USA	AL	Mobile				
aaronto01	1939	8	5	USA	AL	Mobile	1984	8	16	US
aasedo01	1954	9	8	USA	CA	Orange				
abadan01	1972	8	25	USA	FL	Palm Beach				
abadfe01	1985	12	17	D.R.	La Romana	La Romana				
abadijo01	1854	11	4	USA	PA	Philadelphia	1905	5	17	US
abbated01	1877	4	15	USA	PA	Latrobe	1957	1	6	US
abbeybe01	1869	11	11	USA	VT	Essex	1962	6	11	US
abbeych01	1866	10	14	USA	NE	Falls City	1926	4	27	US
abbetde01	1862	2	16	USA	OH	Badaga	1920	2	12	US

View results

3.3.6 HDFSParquetFileWrapper

Custom wrapper for reading Parquet files stored in HDFS. Its base views need the following parameters:

- **File system URI:** A URI whose scheme and authority identify the file system.
 - **HDFS:** `hdfs://<ip>:<port>`.
 - **Amazon S3:** `s3a://@<bucket>`. For configuring the credentials see **Amazon S3** section.
 - **Azure Data Lake Store:**
`adl://<account name>.azuredatalakestore.net/`
For configuring the credentials see **Azure Data Lake Store** section.
 - **Azure Blob Storage:**
`wasb://<container>@<account>.blob.core.windows.net`
For configuring the credentials see **Azure Blob Storage** section.

! Note

If you enter a literal that contains one of the special characters used to indicate interpolation variables @, \, ^, {, }, you have to escape these characters with \.

E.g if the URI contains @ you have to enter \@.

- **Parquet File Path:** path of the file that we want to read.
- **File name pattern:** If you want this wrapper to only obtain data from some of the files of the directory, you can enter a regular expression that matches the names of these files.
For example, if you want the base view to return the data of all the files with the extension parquet set the File name pattern to `(.*)\.parquet`. Optional.
- **Custom core-site.xml file:** configuration file that overrides the default core parameters. Optional.
- **Custom hdfs-site.xml file:** configuration file that overrides the default HDFS parameters. Optional.

File system URI	<input type="text" value="hdfs://quickstart.cloudera:8020"/>		
Parquet File path	<input type="text" value="/user/cloudera/parquet/complex_types.snappy.parquet"/>		
File name pattern	<input type="text"/>		
Custom core-site.xml file	<input type="text" value="None"/>	<input type="button" value="v"/>	<input type="button" value="Configure"/>
Custom hdfs-site.xml file	<input type="text" value="None"/>	<input type="button" value="v"/>	<input type="button" value="Configure"/>
	<input type="checkbox"/> Kerberos enabled		
Kerberos principal name	<input type="text"/>		
Kerberos keytab file	<input type="text" value="None"/>	<input type="button" value="v"/>	<input type="button" value="Configure"/>
Kerberos password	<input type="text"/>		
Kerberos Distribution Center	<input type="text"/>		

HDFSParquetWrapper base view edition

View schema Metadata

View name: parquet

<input type="checkbox"/>	PK	Field Name	Field Type	Description
<input type="checkbox"/>		statecode	text	
<input type="checkbox"/>		countrycode	text	
<input type="checkbox"/>		sitenum	text	
<input type="checkbox"/>		paramcode	text	
<input type="checkbox"/>		poc	text	
<input type="checkbox"/>		latitude	text	
<input type="checkbox"/>		longitude	text	
<input type="checkbox"/>		datum	text	
<input type="checkbox"/>		param	text	
<input type="checkbox"/>		datelocal	text	
<input type="checkbox"/>		timelocal	text	
<input type="checkbox"/>		dategmt	text	
<input type="checkbox"/>		timegmt	text	
<input type="checkbox"/>		degrees	double	
<input type="checkbox"/>		uom	text	
<input type="checkbox"/>		mdl	text	
<input type="checkbox"/>		uncert	text	
<input type="checkbox"/>		qual	text	

Set selected as PK

View schema

The execution of the wrapper returns the values contained in the file.

Execute Query Results

Results Execution Trace Stop Refresh Save Query: SELECT * FROM parquet CONTEXT ('18n'=us_pst,'cache_wait_for_load'=true) TRACE

Total rows received: 7027695 (shown 150)

statecode	countrycode	sitenum	paramcode	poc	latitude	longitude	datum	param	datelocal	timelocal	dategmt	timegmt	degrees
"01"	"003"	"0010"	"44201"	1	30.497478	-87.880258	"NAD83"	"Ozone"	"2016-03-01"	"15:00"	"2016-03-01"	"21:00"	0.041
"01"	"003"	"0010"	"44201"	1	30.497478	-87.880258	"NAD83"	"Ozone"	"2016-03-01"	"16:00"	"2016-03-01"	"22:00"	0.041
"01"	"003"	"0010"	"44201"	1	30.497478	-87.880258	"NAD83"	"Ozone"	"2016-03-01"	"17:00"	"2016-03-01"	"23:00"	0.042
"01"	"003"	"0010"	"44201"	1	30.497478	-87.880258	"NAD83"	"Ozone"	"2016-03-01"	"18:00"	"2016-03-02"	"00:00"	0.041
"01"	"003"	"0010"	"44201"	1	30.497478	-87.880258	"NAD83"	"Ozone"	"2016-03-01"	"19:00"	"2016-03-02"	"01:00"	0.038

View results

4 AMAZON S3

The HDFS Custom Wrapper can access data stored in the following AWS filesystems:

- S3. Note that S3 is being phased out. Use S3N or S3A instead.
- S3N
- S3A. S3A client can read all files created by S3N. It should be used wherever possible.

4.1 CONFIGURING S3 AUTHENTICATION PROPERTIES

Place the credentials in the wrapper configuration file `Custom core-site.xml`:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>
  <name>fs.s3.awsAccessKeyId</name>
  <description>AWS access key ID</description>
  <value>YOUR ACCESS KEY ID</value>
</property>

<property>
  <name>fs.s3.awsSecretAccessKey</name>
  <description>AWS secret key</description>
  <value>YOUR SECRET ACCESS KEY</value>
</property>

</configuration>
```

Alternatively, you could place the credentials in the URI `s3://ID:SECRET@BUCKET/` but this method is discouraged as they will end up in logs and error messages that untrusted people could read.

4.2 CONFIGURING S3N AUTHENTICATION PROPERTIES

Place the credentials in the wrapper configuration file `Custom core-site.xml`:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
```

```
<configuration>

<property>
  <name>fs.s3n.awsAccessKeyId</name>
  <description>AWS access key ID</description>
  <value>YOUR ACCESS KEY ID</value>
</property>

<property>
  <name>fs.s3n.awsSecretAccessKey</name>
  <description>AWS secret key</description>
  <value>YOUR SECRET ACCESS KEY</value>
</property>

</configuration>
```

Alternatively, you could place the credentials in the URI `s3n://ID:SECRET@BUCKET/` but this method is discouraged as they will end up in logs and error messages that untrusted people could read.

4.3 CONFIGURING S3A AUTHENTICATION PROPERTIES

Place the credentials in the wrapper configuration file `Custom core-site.xml`:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>
  <name>fs.s3a.access.key</name>
  <description>AWS access key ID.</description>
  <value>YOUR ACCESS KEY ID</value>
</property>

<property>
  <name>fs.s3a.secret.key</name>
  <description>AWS secret key.</description>
  <value>YOUR SECRET ACCESS KEY</value>
</property>

</configuration>
```

Alternatively, you could place the credentials in the URI `s3a://ID:SECRET@BUCKET/` but this method is discouraged as they will end up in logs and error messages that untrusted people could read.

4.4 SIGNATURE VERSION 4 SUPPORT

When the V4 signing protocol is used, AWS requires the explicit region endpoint to be used —hence **S3A** must be configured to use the specific endpoint. This is done in the configuration option `fs.s3a.endpoint` in the `Custom core-site.xml` of the wrapper. Otherwise a Bad Request exception could be thrown.

As an example of configuration, the endpoint for S3 Frankfurt is `s3.eu-central-1.amazonaws.com`:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
<property>
  <name>fs.s3a.endpoint</name>
  <value>s3.eu-central-1.amazonaws.com</value>
</property>
</configuration>
```

You can find the full list of supported versions for AWS Regions in their website: [Amazon Simple Storage Service \(Amazon S3\)](#).

5 AZURE DATA LAKE STORE

The HDFS Custom Wrapper can access data stored in Azure Data Lake Store.

5.1 CONFIGURING AUTHENTICATION PROPERTIES

Place the credentials in the wrapper configuration file `CUSTOM core-site.xml`:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

  <configuration>
    <property>
      <name>dfs.adls.oauth2.access.token.provider.type</name>
      <value>ClientCredential</value>
    </property>
    <property>
      <name>dfs.adls.oauth2.refresh.url</name>
      <value>YOUR TOKEN ENDPOINT</value>
    </property>
    <property>
      <name>dfs.adls.oauth2.client.id</name>
      <value>YOUR CLIENT ID</value>
    </property>
    <property>
      <name>dfs.adls.oauth2.credential</name>
      <value>YOUR CLIENT SECRET</value>
    </property>
  </configuration>
```

6 AZURE BLOB STORAGE

The HDFS Custom Wrapper can access data stored in Azure Blob Storage.

6.1 CONFIGURING AUTHENTICATION PROPERTIES

Place the credentials in the wrapper configuration file `CUSTOM core-site.xml`:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

  <property>
    <name>fs.azure.account.key.<account>.blob.core.windows.net</name>
    <value>YOUR ACCESS KEY</value>
  </property>

</configuration>
```

7 HDFS COMPRESSED FILES

Hadoop is intended for storing large data volumes, so compression becomes a mandatory requirement here. There are different compression formats available like zlib, bzip2, snappy, LZO and LZ4.

Hadoop has native implementations of compression libraries for performance reasons and for non-availability of Java implementations:

Compression format	Java implementation	Native implementation
DEFLATE	Yes	Yes
gzip	Yes	Yes
bzip2	Yes	No
LZO	No	Yes
Snappy	No	Yes

Compression library implementations

For reading HDFS compressed files using the HDFS Custom Wrapper there are two options:

- Use the Java implementation. In this case the HDFS Custom Wrapper handles compressed files transparently.
- Use the native implementation:
 - for performance reasons or
 - for non-availability of Java implementation

In this case HDFS Custom Wrapper must have Hadoop the native libraries in the `java.library.path`.

8 SECURE CLUSTER WITH KERBEROS

The configuration required for accessing a Hadoop cluster with Kerberos enabled is the same as the one needed to access HDFS and, additionally, the user must supply the Kerberos credentials.

The Kerberos parameters are:

- **Kerberos enabled:** Check it when accessing a Hadoop cluster with Kerberos enabled. Required.
- **Kerberos principal name:** Kerberos v5 Principal name to access HDFS, e.g. `primary/instance@realm`. Optional.
- **Kerberos keytab file:** Keytab file containing the key of the Kerberos principal. Optional.
- **Kerberos password:** Password associated with the principal. Optional.
- **Kerberos Distribution Center:** Kerberos Key Distribution Center. Optional.

The HDFS Custom Wrapper provides **three ways** for accessing a kerberized Hadoop cluster:

1. The client has a valid Kerberos ticket in the **ticket cache** obtained, for example, using the `kinit` command in the Kerberos Client.
In this case only the `kerberos enabled` parameter should be checked. The HDFS wrapper would use the Kerberos ticket to authenticate itself against the Hadoop cluster.
2. The client does not have a valid Kerberos ticket in the ticket cache. In this case you should provide the `kerberos principal name` parameter and
 - 2.1. `kerberos keytab file` parameter or
 - 2.2. `kerberos password` parameter.

In all these **three scenarios** the **krb5.conf** file should be present in the file system. Below there is an example of the Kerberos configuration file:

```
[libdefaults]
renew_lifetime = 7d
forwardable = true
default_realm = EXAMPLE.COM
ticket_lifetime = 24h
dns_lookup_realm = false
dns_lookup_kdc = false

[domain_realm]
sandbox.hortonworks.com = EXAMPLE.COM
cloudera = CLOUDERA

[realms]
```

```
EXAMPLE.COM = {
  admin_server = sandbox.hortonworks.com
  kdc = sandbox.hortonworks.com
}
```

```
CLOUDERA = {
  kdc = quickstart.cloudera
  admin_server = quickstart.cloudera
  max_renewable_life = 7d 0h 0m 0s
  default_principal_flags = +renewable
}
```




[logging]

```
default = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log
kdc = FILE:/var/log/krb5kdc.log
```

The algorithm to locate the `krb5.conf` file is the following:

- If the system property `java.security.krb5.conf` is set, its value is assumed to specify the path and file name.
- If that system property value is not set, then the configuration file is looked for in the directory
 - `<java-home>\lib\security` (Windows)
 - `<java-home>/lib/security` (Solaris and Linux)
- If the file is still not found, then an attempt is made to locate it as follows:
 - `/etc/krb5/krb5.conf` (Solaris)
 - `c:\winnt\krb5.ini` (Windows)
 - `/etc/krb5.conf` (Linux)

There is an **exception**. If you are planning to create HDFS views that use the **same Key Distribution Center and the same realm** the Kerberos Distribution Center parameter can be provided instead of having the `krb5.conf` file in the file system.

File system URI	<input type="text" value="hdfs://quickstart.cloudera:8020"/>
Path	<input type="text" value="/user/hive/warehouse/sample_07/sample_07.csv"/>
	<input type="checkbox"/> Delete after reading
Custom core-site.xml file	<input type="text" value="Local"/> 
	C:/Work/hdfs-site.xml
Custom hdfs-site.xml file	<input type="text" value="None"/> 
Separator	<input type="text"/>
Quote	<input type="text"/>
Comment marker	<input type="text"/>
Escape	<input type="text"/>
	<input type="checkbox"/> Ignore spaces
	<input type="checkbox"/> Header
	<input checked="" type="checkbox"/> Kerberos enabled
Kerberos principal name	<input type="text" value="cloudera-scm/admin\@CLOUDERA"/>
Kerberos keytab file	<input type="text" value="None"/> 
Kerberos password	<input type="password" value="•••••"/>
Kerberos Distribution Center	<input type="text" value="quickstart.cloudera"/>

View edition

9 TROUBLESHOOTING

Symptom

Error message: "Server IPC version X cannot communicate with client version Y".

Resolution

It is a version mismatch problem. Hadoop server version is **newer** than the version distributed in the custom wrapper artifact `denodo-hdfs-customwrapper-<VERSION>-jar-with-dependencies`.

To solve the problem you should use the custom wrapper artifact `denodo-hdfs-customwrapper-<VERSION>` and copy the Hadoop server libraries to the `$DENODO_PLATFORM_HOME/extensions/thirdparty/lib` directory.

Symptom

Error message: "SIMPLE authentication is not enabled. Available:[TOKEN, KERBEROS]".

Resolution

You are trying to connect to a Kerberos-enabled Hadoop cluster. You should configure the custom wrapper accordingly. See [Secure cluster with Kerberos section](#) for **configuring Kerberos** on this custom wrapper.

Symptom

Error message: "Cannot get Kerberos service ticket: KrbException: Server not found in Kerberos database (7)".

Resolution

Check that `nslookup` is returning the fully qualified hostname of the KDC. If not, modify the `/etc/hosts` of the client machine for the KDC entry to be of the form "IP address fully.qualified.hostname alias".

Symptom

Error message: "Invalid hostname in URI s3n://<id>:<secret>@<bucket>".

Resolution

Check your bucket name: underscores are not permitted.

Also check your secret key, if it contains "/" and "+" symbols they need to be encoded in the URL. This method of **placing credentials in the URL is discouraged**. Configure the credentials on the core-site.xml instead (see **Amazon S3 support** section).

Symptom

```
Error message: "Error accessing Parquet file: Could not read footer:
java.io.IOException: Could not read footer for file FileStatus{path=
hdfs://serverhdfs/apps/hive/warehouse/parquet/.hive-staging_hive_2017-03-
06_08-/-ext-10000; isDirectory=true; modification_time=1488790684826;
access_time=0; owner=hive; group=hdfs; permission=rwxr-xr-x;
isSymlink=false}"
```

Resolution

Hive could store metadata into a parquet file folder. You can check in the error message, if the custom wrapper is trying to access to any metadata. In the error of the example you can see that it is accessing a folder called .hive-staging*. The solution is to configure Hive to store metadata in other location.

Symptom

Error message: "Could not initialize class org.xerial.snappy.Snappy"

Resolution

On Linux platforms, an error may occur when Snappy compression/decompression is enabled although its library is available from the classpath.

The native library snappy-<version>-libsnappyjava.so for Snappy compression is included in the snappy-java-<version>.jar file. When the JVM initializes the JAR, the library is added to the default temp directory. If the default temp directory is mounted with a noexec option, it results in the above exception.

One solution is to specify a different temp directory that has already been mounted without the noexec option, as follows:

```
-Dorg.xerial.snappy.tmpdir=/path/to/newtmp
```

10 APPENDICES

10.1 HOW TO USE THE HADOOP VENDOR'S CLIENT LIBRARIES

In some cases, it is advisable to use the libraries of the Hadoop vendor you are connecting to (Cloudera, Hortonworks, ...), instead of the Apache Hadoop libraries distributed in the HDFS Custom Wrapper.

In order to use the Hadoop vendor libraries there is no need to import the HDFS Custom Wrapper as an extension as it is explained in the **Importing the custom wrapper into VDP** section.

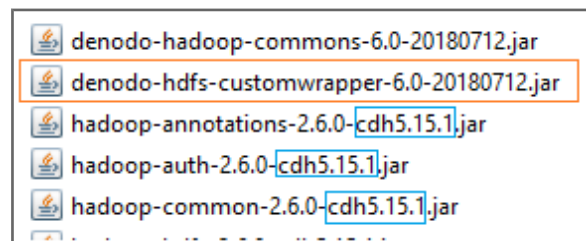
You have to create the HDFS data sources using the **'Classpath' parameter** instead of the 'Select Jars' option.

Click Browse to select the directory containing the required dependencies for the HDFS Custom Wrapper, that is:

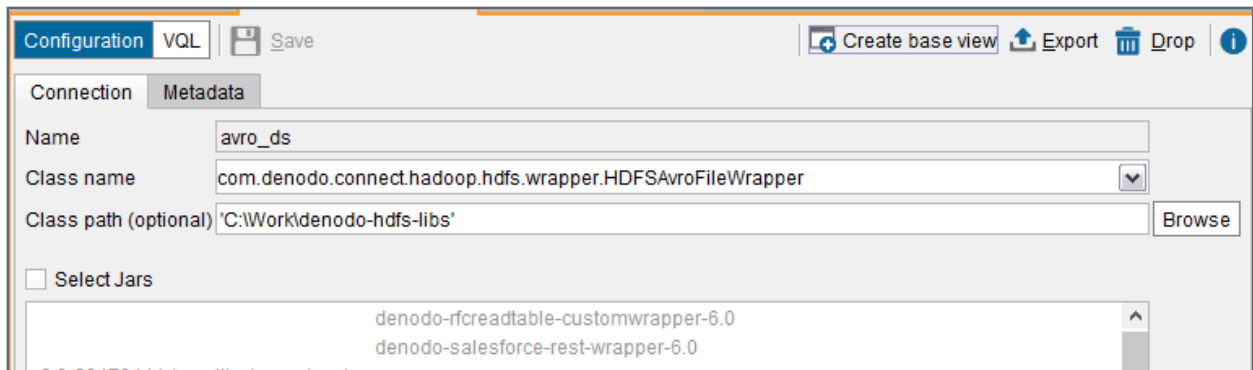
- The `denodo-hdfs-customwrapper-${version}.jar` file of the `dist` directory of the HDFS Custom Wrapper distribution (highlighted in orange in the image below).
- The contents of the `lib` directory of the HDFS Custom Wrapper distribution, replacing the Apache Hadoop libraries with the vendor specific ones (highlighted in blue in the image below, the suffix indicating that they are Cloudera jars).

Here you can find the libraries for Cloudera and Hortonworks Hadoop distributions:

- Cloudera repository: <https://repository.cloudera.com/artifactory/cloudera-repos/org/apache/hadoop/>
- Hortonworks repository: <http://repo.hortonworks.com/content/repositories/releases/org/apache/hadoop/>



C:\Work\denodo-hdfs-libs directory



Configuration VQL Save Create base view Export Drop

Connection Metadata

Name avro_ds

Class name com.denodo.connect.hadoop.hdfs.wrapper.HDFSAvroFileWrapper

Class path (optional) 'C:\Work\denodo-hdfs-libs' Browse

☐ Select Jars

denodo-rfcreadtable-customwrapper-6.0

denodo-salesforce-rest-wrapper-6.0

HDFS Data Source

! Note

When clicking **Browse**, you will browse the file system of the host where the Server is running and not where the Administration Tool is running.