Hortonworks Data Platform

Data Integration Services with HDP

(Jun 12, 2013)

docs.hortonworks.com

Hortonworks Data Platform : Data Integration Services with HDP

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, Zookeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

Table of Contents

List of Tables

1. Using Data Integration Services Powered by Talend

Talend Open Studio for Big Data is a powerful and versatile open source data integration solution. It enables an enterprise to work with existing data and existing systems, and use Hadoop to power large scale data analysis across the enterprise.

Talend Open Studio (TOS) is distributed as an add-on for Hortonworks Data Platform (HDP). TOS uses the following HDP components:

- Enable users to read/write from/to Hadoop as a data source/sink.
- HCatalog Metadata services enable users to import raw data into Hadoop (HBase and HDFS), create, and manage schemas.
- Pig and Hive for analyzing these data sets.
- Enable users to schedule these ETL jobs on a recurring basis on a Hadoop Cluster using Oozie.

This document includes the following sections:

- Prerequisites
- Instructions
  - Deploying Talend Open Studio
  - Writing Talend job for data import
  - Modifying the job to perform data analysis

For more information on Talend Open Studio, see Talend Open Studio v5.3 Documentation.

1.1. Prerequisites

- Ensure that you have deployed HDP for all the nodes in your cluster. For instructions on deploying HDP, see "Getting Ready to Install" in *Installing HDP Using Apache Ambari* here.
- Ensure that you create a home directory for the user launching the TOS in the HDFS cluster.

  EXAMPLE: If `hdptestuser` is responsible for launching TOS, then execute the following command on the gateway machine as the administrator user (HDFS user) to create home directory:

  ```
  % hadoop dfs –mkdir /user/hdptestuser
  ```

- Ensure the user launching the TOS has appropriate permissions on the HDP cluster.

EXAMPLE: If `hdptestuser` is responsible for launching TOS, then execute the following command on the gateway machine as the administrator user (HDFS user) to provide the required permissions:

```
% hadoop dfs –chown hdptestuser:hdptestuser /user/hdptestuser
```

1.2. Instructions

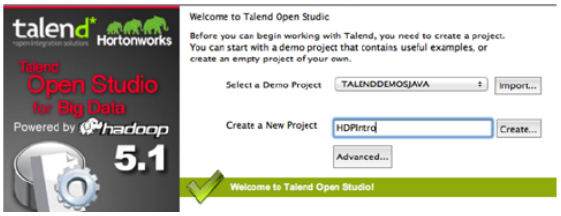
This section provides you instructions on the following:

- Deploying Talend Open Studio
- Writing Talend job for data import
- Modifying the job to perform data analysis

1.2.1. Deploying Talend Open Studio

Use the following instructions to set-up Talend Open Studio:

1. Download and launch the application.
   a. Download the Talend Open Studio add-on for HDP from here.
   b. After the download is complete, unzip the contents in an install location.
   c. Invoke the executable file corresponding to your operating system.
   d. Read and accept the end user license agreement.
2. Create a new project.
   a. Provide a project name (for example, HDPIntro) and click the **Create** button.



   b. Click **Finish** on the **New Project** dialog.
   c. Select the newly created project and click **Open**.
   d. The **Connect To TalendForge** dialog appears, you can choose to register or click **Skip** to continue.
   e. You should now see the progress information bar and a welcome window. Wait for the application to initialize and then click **Start now!** to continue. Talend Open Studio (TOS) main window appears and is now ready for use.

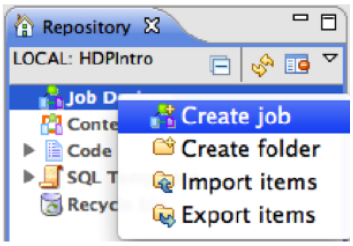
1.2.2. Writing Talend job for data import

This section provides instructions on designing a simple job of importing a file into the Hadoop cluster.

1. Create a new job.

   a. In the Repository tree view, right-click the "Job Designs" node.

   b. From the contextual menu, select "Create job".



   c. In the "New Job" wizard provide a name (for example HDPJob) and click "Finish".

   d. An empty design workspace corresponding to the Job name opens up.

2. Create a sample input file.

   a. Under the `/tmp` directory of your TOS master deployment machine, create a text file (for example: `input.txt`) with the following contents:

```
101;Adam;Wiley;Sales
102;Brian;Chester;Service
103;Julian;Cross;Sales
104;Dylan;Moore;Marketing
105;Chris;Murphy;Service
106;Brian;Collingwood;Service
107;Michael;Muster;Marketing
108;Miley;Rhodes;Sales
109;Chris;Coughlan;Sales
110;Aaron;King;Marketing
```

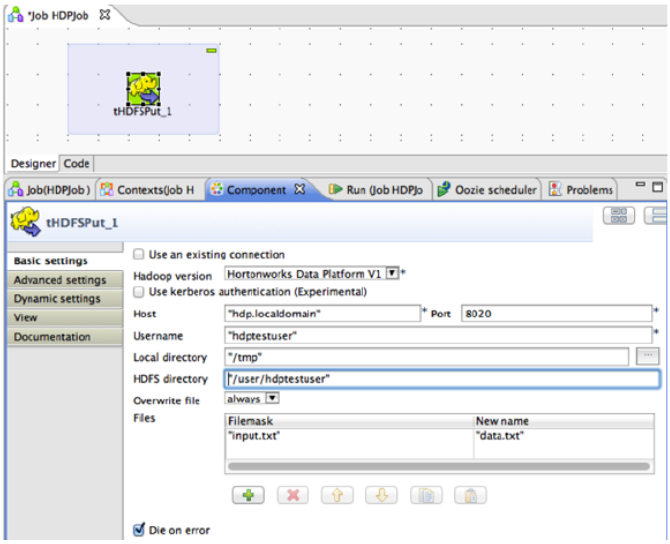
3. Build the job. Jobs are composed of components that are available in the Palette.

a. Expand the **Big Data** tab in the Palette.

b. Click on the component `tHDFSPut` and click on the design workspace to drop this component.

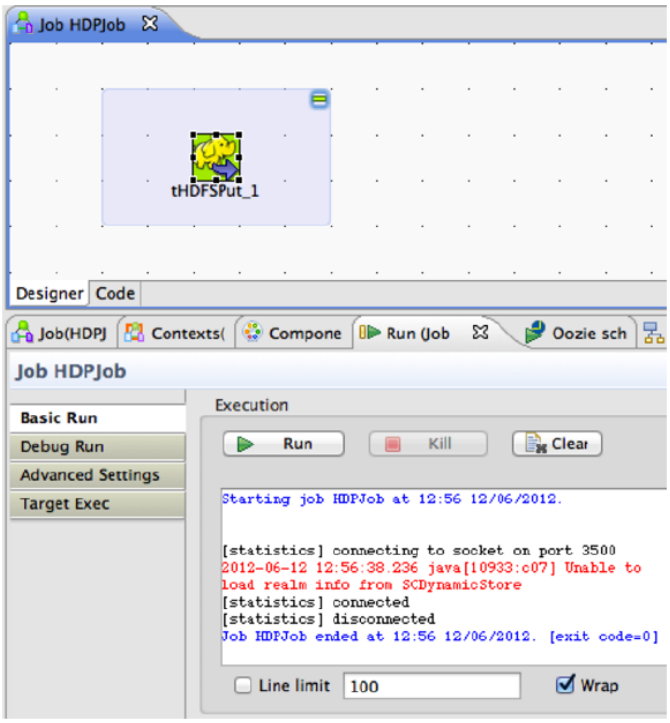
c. Double-click **tHDFSPut** to define component in its **Basic Settings** view.

d. Set the values in the Basic Settings corresponding to your HDP cluster (see the screenshot given below):



4. Run the job. You now have a working job. You can run it by clicking the green play icon.

   You should see the following:



5. Verify the import operation. From the gateway machine or the HDFS client, open a console window and execute the following command:

```
hadoop dfs -ls /user/testuser/data.txt
```

You should see the following result on your terminal window:

```
Found 1 items
-rw-r--r-- 3 testuser testuser
252 2012-06-12 12:52 /user/
testuser/data.txt
```

This message indicates that the local file was successfully created in your Hadoop cluster.

1.2.3. Modifying the job to perform data analysis

Use the following instructions to aggregate data using Apache Pig.

1. Add Pig component from the Big Data Palette.

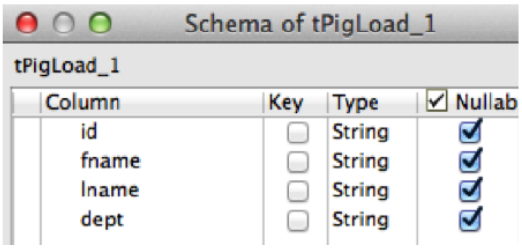
   a. Expand the **Pig** tab in the Big Data Palette.

   b. Click on the component **tPigLoad** and place it in the design workspace.

2. Define basic settings for Pig component.

   a. Double-click **tPigLoad** component to define its Basic Settings.

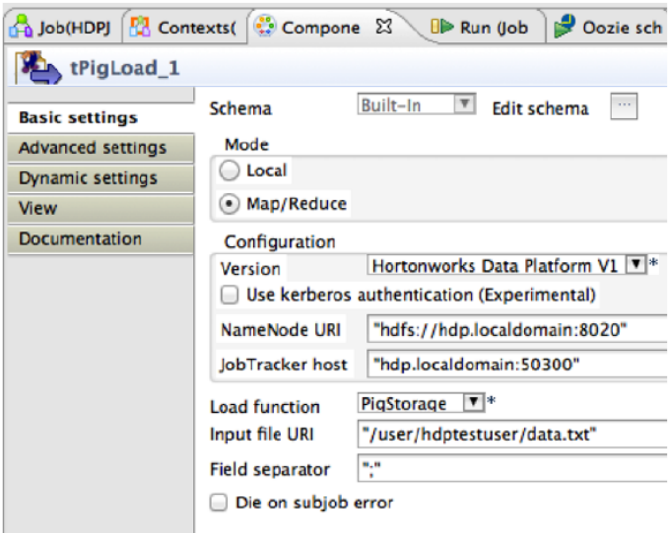
b. Click the **Edit Schema** button ("**...**" button). Define the schema of the input data as shown below and click **OK**:



c. Provide the values for the mode, configuration, NameNode URI, JobTracker host, load function, and input file URI fields as shown.

Important

Ensure that the NameNode URI and the JobTracker host corresponds to accurate values available in your HDP cluster. The Input File URI corresponds to the path of the previously imported `input.txt` file.

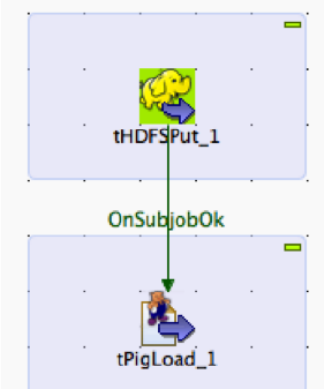


3. Connect the Pig and HDFS components to define the workflow.

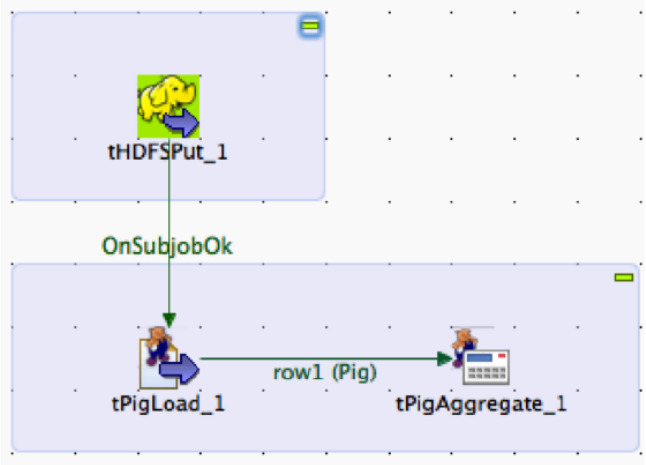
   a. Right-click the source component (`tHDFSPut`) on your design workspace.

   b. From the contextual menu, select Trigger -> On Subjob Ok.

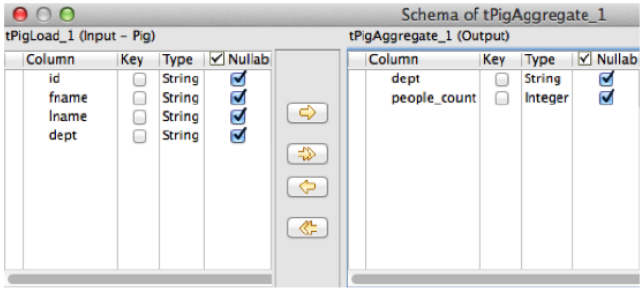
   c. Click the target component (`tPigLoad`).



4. Add and connect Pig aggregate component.
   a. Add the component **tPigAggregate** next to **tPigLoad**.
   b. From the contextual menu, right-click on **tPigLoad** and select **-> Pig Combine**.
   c. Click on **tPigAggregate**.



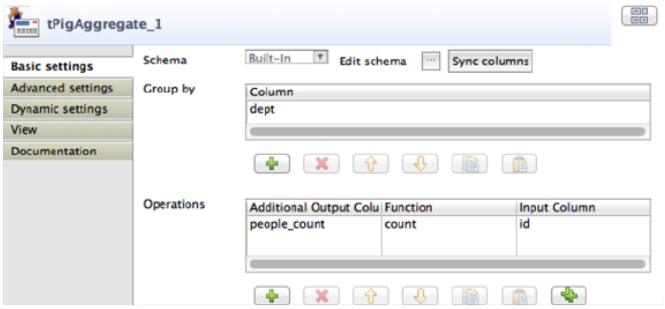
5. Define basic settings for Pig Aggregate component.
   a. Double-click **tPigAggregate** to define the component in its Basic Settings.
   b. Click on the "Edit schema" button and define the output schema as shown below:



6. Define aggregation function for the data.

   a. Add a column to `Group by`, choose `dept`.

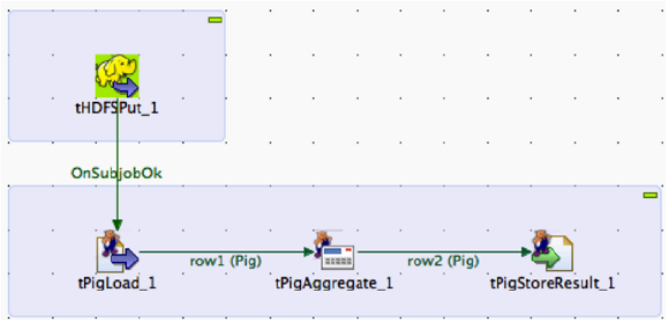
   b. In the Operations table, choose the `people_count` in the Additional Output column, function as `count` and input column `id` as shown:



7. Add and connect Pig data storage component

   a. Add the component `tPigStoreResult` next to `tPigAggregate`.

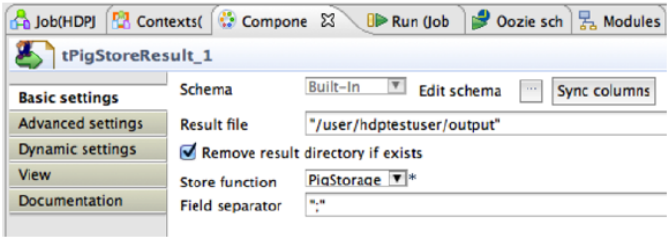
   b. From the contextual menu, right-click on `tPigLoad`, select Row -> Pig Combine and click on `tPigStoreResult`.



8. Define basic settings for data storage component.

   a. Double-click **tPigStoreResult** to define the component in its Basic Settings view.

   b. Specify the result directory on HDFS as shown:



9. Run the modified Talend job. The modified Talend job is ready for execution.

   Save the job and click the play icon to run as instructed in Step 4.

10. Verify the results.

    a. From the gateway machine or the HDFS client, open a console window and execute the following command:

    ```
    hadoop dfs -cat /user/testuser/output/part-r-00000
    ```

    b. You should see the following output:

    ```
    Sales;4
    Service;3
    Marketing;3
    ```

2. Using HDP for Metadata Services (HCatalog)

Hortonworks Data Platform deploys Apache HCatalog to manage the metadata services for your Hadoop cluster. Apache HCatalog is a table and storage management service for data created using Apache Hadoop. This includes:

- Providing a shared schema and data type mechanism.
- Providing a table abstraction so that users need not be concerned with where or how their data is stored.
- Providing interoperability across data processing tools such as Pig, MapReduce, and Hive.

Note

HCatalog 0.5.0 was the final version released from the Apache Incubator. In March 2013, HCatalog graduated from the Apache Incubator and became part of the Apache Hive project. New releases of Hive include HCatalog, starting with Hive 0.11.0.

HCatalog includes two documentation sets:

1. General information about HCatalog

   This documentation covers installation and user features. The section Using HCatalog provides links to individual documents in the HCatalog documentation set.

2. WebHCat information

   WebHCat is a web API for HCatalog and related Hadoop components. The section Using WebHCat provides links to user and reference documents, and includes a technical update about standard WebHCat parameters.

For more details on the Apache Hive project, including HCatalog and WebHCat, see Using Apache Hive and the following resources:

- Hive project home page
- Hive wiki home page
- Hive mailing lists

2.1. Using HCatalog

For details about HCatalog, see the following resources in the HCatalog documentation set:

- HCatalog Overview
- Installation From Tarball
- Load and Store Interfaces

- Input and Output Interfaces
- Reader and Writer Interfaces
- Command Line Interface
- Storage Formats
- Dynamic Partitioning
- Notification
- Storage Based Authorization

2.2. Using WebHCat

WebHCat provides a REST API for HCatalog and related Hadoop components.

Note

The original work to add REST APIs to HCatalog was called Templeton. For backward compatibility the name still appears in URLs, log file names, etc.

For details about WebHCat (Templeton), see the following resources:

- Overview
- Installation
- Configuration
- **Reference**
  - Resource List
  - GET :version
  - GET status
  - GET version
  - DDL Resources: Summary and Commands
  - POST mapreduce/streaming
  - POST mapreduce/jar
  - POST pig
  - POST hive
  - GET queue
  - GET queue/:jobid

- DELETE queue/:jobid

2.2.1. Technical Update: WebHCat Standard Parameters

The "Security" section of the WebHCat Overview should be updated with information in the Note below:

2.2.1.1. Security

The current version supports two types of security:

- Default security (without additional authentication)
- Authentication via Kerberos

2.2.1.1.1. Standard Parameters

Every REST resource can accept the following parameters to aid in authentication:

- user.name: The user name as a string. Only valid when using default security.
- SPNEGO credentials: When running with Kerberos authentication.

Note

The user.name parameter is part of POST parameters for POST calls, and part of the URL for other calls.

For example, to specify user.name in a GET :table command:

```
% curl -s 'http://localhost:50111/templeton/v1/ddl/database/default/
table/my_table?user.name=ctdean'
```

And to specify user.name in a POST :table command:

```
% curl -s -d user.name=ctdean \
       -d rename=test_table_2 \
       'http://localhost:50111/templeton/v1/ddl/database/default/
table/test_table'
```

2.2.1.1.2. Security Error Response

If the user.name parameter is not supplied when required, the following error will be returned:

```
{
  "error": "No user found.  Missing user.name parameter."
}
```

3. Using Apache Hive

Hortonworks Data Platform deploys Apache Hive for your Hadoop cluster.

Hive is a data warehouse infrastructure built on top of Hadoop. It provides tools to enable easy data ETL, a mechanism to put structures on the data, and the capability for querying and analysis of large data sets stored in Hadoop files.

Hive provides SQL on Hadoop, enabling users familiar with SQL to query the data. At the same time, Hive's SQL allows programmers who are familiar with the MapReduce framework to plug in their custom mappers and reducers to perform more sophisticated analysis that may not be supported by the built-in capabilities of the language.

Hive now includes the HCatalog subproject for managing metadata services on your Hadoop cluster. See Using HDP for Metadata Services (HCatalog) for more information.

Hive Documentation

Documentation for Hive release 0.11 can be found in the Hive wiki and javadocs.

The Hive wiki contains documentation organized in these sections:

- General Information about Hive
- User Documentation
- Administrator Documentation
- Resources for Contributors

Javadocs describe the Hive API.

Hive JIRAs

Issue tracking for Hive bugs and improvements can be found here: Hive JIRAs.

Hive ODBC Driver

Hortonworks provides a Hive ODBC driver that allows you to connect popular Business Intelligence (BI) tools to query, analyze and visualize data stored within the Hortonworks Data Platform.

- Download the Hortonworks Hive ODBC driver for Linux, Windows, or Mac OS X from the "Add-Ons" for Hortonworks Data Platform 1.3 here.
- The instructions on installing and using this driver are available here.

Hive Metastore Scripts

Metastore database initialization and upgrade scripts for Hive 0.11 are exactly the same as those for Hive 0.10, because the schema did not change. Script names were not changed to match the new release number.

For example, the script "`hive-schema-0.10.0.mysql.sql`" initializes a MySQL database for the Hive 0.11 metastore. The next section contains an example of initializing

a Postgres database for the Hive 0.11 metastore with a script that has "0.10.0" in the filename.

Using Postgres for the Hive Metastore

If you use a PostgreSQL (Postgres) database for the Hive metastore, follow these setup guidelines:

- Add this property to the `hive-site.xml` file:

```
<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>false</value>
</property>
```

- Initialize the metastore database schema using the bundled metastore schema initialization script found at `/usr/lib/hive/scripts/metastore/upgrade/postgres/hive-schema-0.10.0.postgres.sql`.

- **To upgrade:** For Postgres 9.1 or above, add this parameter to the `/var/lib/pgsql/data/postgresql.conf` file:

```
standard_conforming_strings = off
```

  This setting will only be required for upgrading from the current deployment, which uses Postgres 8.4. The default behavior changes in Postgres 9.1.

4. Using HDP for Workflow and Scheduling (Oozie)

Hortonworks Data Platform deploys Apache Oozie for your Hadoop cluster.

Oozie is a server-based workflow engine specialized in running workflow jobs with actions that execute Hadoop jobs, such as MapReduce, Pig, Hive, Sqoop, HDFS operations, and sub-workflows. Oozie supports coordinator jobs, which are sequences of workflow jobs that are created at a given frequency and start when all of the required input data is available. A command-line client and a browser interface allow you to manage and administer Oozie jobs locally or remotely.

For additional Oozie documentation, use the following resources:

- Quick Start Guide
- Developer Documentation
  - Oozie Workflow Overview
  - Running the Examples
  - Workflow Functional Specification
  - Coordinator Functional Specification
  - Bundle Functional Specification
  - EL Expression Language Quick Reference
  - Command Line Tool
  - Workflow Rerun
  - Email Action
  - Writing a Custom Action Executor
  - Oozie Client Javadocs
  - Oozie Core Javadocs
  - Oozie Web Services API
- Administrator Documentation
  - Oozie Installation and Configuration
  - Oozie Monitoring
  - Command Line Tool

5. Using Apache Sqoop

Hortonworks Data Platform deploys Apache Sqoop for your Hadoop cluster.

Sqoop is a tool designed to transfer data between Hadoop and relational databases. You can use Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS), transform the data in Hadoop MapReduce, and then export the data back into an RDBMS. Sqoop automates most of this process, relying on the database to describe the schema for the data to be imported. Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance.

This document includes the following sections:

- Apache Sqoop Connectors
- Sqoop Import Table Commands
- Netezza Connector
- Sqoop-HCatalog Integration

For additional information see the Sqoop documentation, including these sections in the User Guide:

- Basic Usage
- Sqoop Tools
- Troubleshooting

5.1. Apache Sqoop Connectors

Sqoop uses a connector based architecture which supports plugins that provide connectivity to new external systems. Using specialized connectors, Sqoop can connect with external systems that have optimized import and export facilities, or do not support native JDBC. Connectors are plugin components based on Sqoop's extension framework and can be added to any existing Sqoop installation.

Hortonworks provides the following connectors for Sqoop:

- **MySQL connector:** This connector is included in the HDP distribution; the instructions on configuring this connector are available here.
- **Netezza connector:** This connector is included in the HDP distribution and installs with Sqoop; see below for more information.
- **Oracle JDBC connector:** The instructions on configuring this connector are available here.
- **Hortonworks Connector for Teradata:** This connector, based on the Teradata Connector for Hadoop, can be downloaded from here.

A Sqoop connector for SQL Server is available from Microsoft:

- **SQL Server R2 connector:** This connector and its documentation can be downloaded from here.

5.2. Sqoop Import Table Commands

When connecting to an Oracle database, the Sqoop `import` command requires case-sensitive table names and usernames (typically uppercase). Otherwise the import fails with error message "Attempted to generate class with no columns!"

Prior to the resolution of SQOOP-741, `import-all-tables` would fail for an Oracle database. See the JIRA for more information.

The `import-all-tables` command has additional restrictions. See Chapter 8 in the Sqoop User Guide.

5.3. Netezza Connector

Netezza connector for Sqoop is an implementation of the Sqoop connector interfaces for accessing a Netezza data warehouse appliance, so that data can be exported and imported to a Hadoop environment from Netezza data warehousing environments.

The HDP Sqoop distribution includes Netezza connector software. To deploy it, the only requirement is that you acquire the JDBC jar file (named `nzjdbc.jar`) from IBM and copy it to the `/usr/local/nz/lib` directory.

5.3.1. Extra Arguments

This table describes extra arguments supported by the Netezza connector:

Table 5.1. Supported Netezza Extra Arguments

Argument	Description
`--partitioned-access`	Whether each mapper acts on a subset of data slices of a table or all. Default is "false" for standard mode and "true" for direct mode.
`--max-errors`	Applicable only in direct mode. This option specifies the error threshold per mapper while transferring data. If the number of errors encountered exceeds this threshold, the job fails. Default value is 1.
`--log-dir`	Applicable only in direct mode. Specifies the directory where Netezza external table operation logs are stored. Default value is /tmp.

5.3.2. Direct Mode

Netezza connector supports an optimized data transfer facility using the Netezza external tables feature. Each map task of Netezza connector's import job works on a subset of the Netezza partitions and transparently creates and uses an external table to transport data. Similarly, export jobs use the external table to push data fast onto the NZ system. Direct mode does not support staging tables, upsert options, etc.

Direct mode is specified by the `--direct` Sqoop option.

Here is an example of a complete command line for import using the Netezza external table feature.

```
$ sqoop import \
    --direct \
    --connect jdbc:netezza://nzhost:5480/sqoop \
    --table nztable \
    --username nzuser \
    --password nzpass \
    --target-dir hdfsdir
```

Here is an example of a complete command line for export with tab (\t) as the field terminator character.

```
$ sqoop export \
    --direct \
    --connect jdbc:netezza://nzhost:5480/sqoop \
    --table nztable \
    --username nzuser \
    --password nzpass \
    --export-dir hdfsdir \
    --input-fields-terminated-by "\t"
```

5.3.3. Null String Handling

In direct mode the Netezza connector supports the null-string features of Sqoop. Null string values are converted to appropriate external table options during export and import operations.

Table 5.2. Supported Export Control Arguments

Argument	Description
`--input-null-string <null-string>`	The string to be interpreted as null for string columns.
`--input-null-non-string <null-string>`	The string to be interpreted as null for non-string columns.

In direct mode, both the arguments must either be left to the default values or explicitly set to the same value. Furthermore, the null string value is restricted to 0-4 UTF-8 characters.

On export, for non-string columns, if the chosen null value is a valid representation in the column domain, then the column might not be loaded as null. For example, if the null string value is specified as "1", then on export, any occurrence of "1" in the input file will be loaded as value 1 instead of NULL for int columns.

For performance and consistency, specify the null value as an empty string.

Table 5.3. Supported Import Control Arguments

Argument	Description
`--null-string <null-string>`	The string to be interpreted as null for string columns.
`--null-non-string <null-string>`	The string to be interpreted as null for non-string columns.

In direct mode, both the arguments must either be left to the default values or explicitly set to the same value. Furthermore, the null string value is restricted to 0-4 UTF-8 characters.

On import, for non-string columns in the current implementation, the chosen null value representation is ignored for non-character columns. For example, if the null string value is specified as "\N", then on import, any occurrence of NULL for non-char columns in the table will be imported as an empty string instead of \N, the chosen null string representation.

For performance and consistency, specify the null value as an empty string.

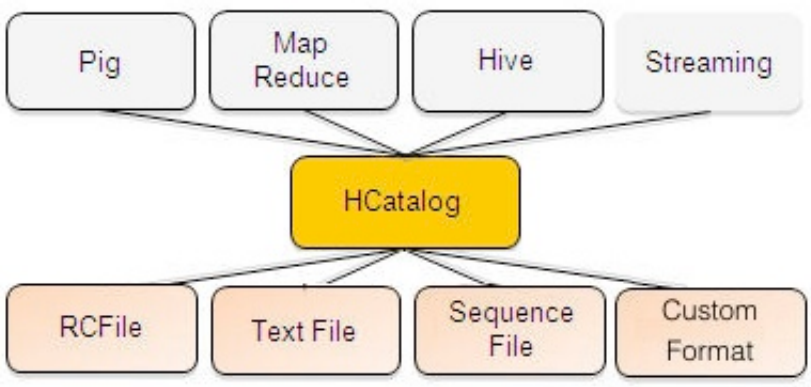
5.4. Sqoop-HCatalog Integration

This section describes the integration of HCatalog with Sqoop, which is introduced in HDP 1.3.0.

5.4.1. HCatalog Background

HCatalog is a table and storage management service for Hadoop that enables users with different data processing tools – Pig, MapReduce, and Hive – to more easily read and write data on the grid. HCatalog's table abstraction presents users with a relational view of data in the Hadoop distributed file system (HDFS) and ensures that users need not worry about where or in what format their data is stored: RCFile format, text files, or SequenceFiles.

HCatalog supports reading and writing files in any format for which a Hive SerDe (serializer-deserializer) has been written. By default, HCatalog supports RCFile, CSV, JSON, and SequenceFile formats. To use a custom format, you must provide the InputFormat and OutputFormat as well as the SerDe.



The ability of HCatalog to abstract various storage formats is used in providing RCFile (and future file types) support to Sqoop.

5.4.2. Exposing HCatalog Tables to Sqoop

HCatalog integration with Sqoop is patterned on an existing feature set that supports Avro and Hive tables. Five new command line options are introduced, and some command line options defined for Hive are reused.

5.4.2.1. New Command Line Options

`--hcatalog-database`	Specifies the database name for the HCatalog table. If not specified, the default database name 'default' is

	used. Providing the `--hcatalog-database` option without `--hcatalog-table` is an error. This is not a required option.
`--hcatalog-table`	The argument value for this option is the HCatalog tablename. The presence of the `--hcatalog-table` option signifies that the import or export job is done using HCatalog tables, and it is a required option for HCatalog jobs.
`--hcatalog-home`	The home directory for the HCatalog installation. The directory is expected to have a lib subdirectory and a share/hcatalog subdirectory with necessary HCatalog libraries. If not specified, the system environment variable HCAT_HOME will be checked and failing that, a system property hcatalog.home will be checked. If none of these are set, the default value will be used and currently the default is set to /usr/lib/hcatalog. This is not a required option.
`--create-hcatalog-table`	This option specifies whether an HCatalog table should be created automatically when importing data. By default, HCatalog tables are assumed to exist. The table name will be the same as the database table name translated to lower case. Further described in Automatic Table Creation below.
`--hcatalog-storage-stanza`	This option specifies the storage stanza to be appended to the table. Further described in Automatic Table Creation below.

5.4.2.2. Supported Sqoop Hive Options

The following Sqoop options are also used along with the `--hcatalog-table` option to provide additional input to the HCatalog jobs. Some of the existing Hive import job options are reused with HCatalog jobs instead of creating HCatalog-specific options for the same purpose.

`--map-column-hive`	This option maps a database column to HCatalog with a specific HCatalog type.
`--hive-home`	The Hive home location.
`--hive-partition-key`	Used for static partitioning filter. The partitioning key should be of type STRING. There can be only one static partitioning key.
`--hive-partition-value`	The value associated with the partition.

5.4.2.3. Unsupported Sqoop Options

Sqoop Hive options that are not supported with HCatalog jobs:

- `--hive-import`
- `--hive-overwrite`

Other Sqoop options that are not supported with HCatalog jobs:

- `--direct`
- `--export-dir`
- `--target-dir`
- `--warehouse-dir`
- `--append`

5.4.2.4. Ignored Sqoop Options

All input delimiter options are ignored.

Output delimiters are generally ignored unless either `--hive-drop-import-delims` or `--hive-delims-replacement` is used. When the `--hive-drop-import-delims` or `--hive-delims-replacement` option is specified, database table columns of type STRING will be post-processed to either remove or replace the delimiters, respectively. (See Delimited Text Formats and Field and Line Delimiter Characters below.) This is only needed if the HCatalog table uses text format.

5.4.3. Automatic Table Creation

One of the key features of Sqoop is to manage and create the table metadata when importing into Hadoop. HCatalog import jobs also provide for this feature with the option `--create-hcatalog-table`. Furthermore, one of the important benefits of the HCatalog integration is to provide storage agnosticism to Sqoop data movement jobs. To provide for that feature, HCatalog import jobs provide an option that lets a user specifiy the storage format for the created table.

The option `--create-hcatalog-table` is used as an indicator that a table has to be created as part of the HCatalog import job.

The option `--hcatalog-storage-stanza` can be used to specify the storage format of the newly created table. The default value for this option is "stored as rcfile". The value specified for this option is assumed to be a valid Hive storage format expression. It will be appended to the CREATE TABLE command generated by the HCatalog import job as part of automatic table creation. Any error in the storage stanza will cause the table creation to fail and the import job will be aborted.

Any additional resources needed to support the storage format referenced in the option `--hcatalog-storage-stanza` should be provided to the job either by placing them in $HIVE_HOME/lib or by providing them in HADOOP_CLASSPATH and LIBJAR files.

If the option `--hive-partition-key` is specified, then the value of this option is used as the partitioning key for the newly created table. Only one partitioning key can be specified with this option.

Object names are mapped to the lowercase equivalents as specified below when mapped to an HCatalog table. This includes the table name (which is the same as the external store table name converted to lower case) and field names.

5.4.4. Delimited Text Formats and Field and Line Delimiter Characters

HCatalog supports delimited text format as one of the table storage formats. But when delimited text is used and the imported data has fields that contain those delimiters, then the data may be parsed into a different number of fields and records by Hive, thereby losing data fidelity.

For this case, one of these existing Sqoop import options can be used:

- `--hive-delims-replacement`
- `--hive-drop-import-delims`

If either of these options is provided on input, then any column of type STRING will be formatted with the Hive delimiter processing and then written to the HCatalog table.

5.4.5. HCatalog Table Requirements

The HCatalog table should be created before using it as part of a Sqoop job if the default table creation options (with optional storage stanza) are not sufficient. All storage formats supported by HCatalog can be used with the creation of the HCatalog tables. This makes this feature readily adopt new storage formats that come into the Hive project, such as ORC files.

5.4.6. Support for Partitioning

The Sqoop HCatalog feature supports the following table types:

- Unpartitioned tables
- Partitioned tables with a static partitioning key specified
- Partitioned tables with dynamic partition keys from the database result set
- Partitioned tables with a combination of a static key and additional dynamic partitioning keys

5.4.7. Schema Mapping

Sqoop currently does not support column name mapping. However, the user is allowed to override the type mapping. Type mapping loosely follows the Hive type mapping already present in Sqoop except that SQL types “FLOAT” and “REAL” are mapped to HCatalog type “float”. In the Sqoop type mapping for Hive, these two are mapped to “double”. Type mapping is primarily used for checking the column definition correctness only and can be overridden with the `--map-column-hive` option.

All types except binary are assignable to a String type.

Any field of number type (int, shortint, tinyint, bigint and bigdecimal, float and double) is assignable to another field of any number type during exports and imports. Depending on the precision and scale of the target type of assignment, truncations can occur.

Furthermore, date/time/timestamps are mapped to string (the full date/time/timestamp representation) or bigint (the number of milliseconds since epoch) during imports and exports.

BLOBs and CLOBs are only supported for imports. The BLOB/CLOB objects when imported are stored in a Sqoop-specific format and knowledge of this format is needed for processing these objects in a Pig/Hive job or another Map Reduce job.

Database column names are mapped to their lowercase equivalents when mapped to the HCatalog fields. Currently, case-sensitive database object names are not supported.

Projection of a set of columns from a table to an HCatalog table or loading to a column projection is allowed (subject to table constraints). The dynamic partitioning columns, if any, must be part of the projection when importing data into HCatalog tables.

Dynamic partitioning fields should be mapped to database columns that are defined with the NOT NULL attribute (although this is not validated). A null value during import for a dynamic partitioning column will abort the Sqoop job.

5.4.8. Support for HCatalog Data Types

All the primitive HCatalog types are supported. Currently all the complex HCatalog types are unsupported.

BLOB/CLOB database types are only supported for imports.

5.4.9. Providing Hive and HCatalog Libraries for the Sqoop Job

With the support for HCatalog added to Sqoop, any HCatalog job depends on a set of jar files being available both on the Sqoop client host and where the Map/Reduce tasks run. To run HCatalog jobs, the environment variable HADOOP_CLASSPATH must be set up as shown below before launching the Sqoop HCatalog jobs.

```
HADOOP_CLASSPATH=$(hcat -classpath)
export HADOOP_CLASSPATH
```

The necessary HCatalog dependencies will be copied to the distributed cache automatically by the Sqoop job.

5.4.10. Examples

- Create an HCatalog table, such as:

```
hcat -e "create table txn(txn_date string, cust_id string, amount float,
 store_id int)
partitioned by (cust_id string) stored as rcfile;"
```

- Then Sqoop import and export of the "txn" HCatalog table can be invoked as follows:

Import

```
$SQOOP_HOME/bin/sqoop import --connect <jdbc-url> -table <table-
name> --hcatalog-table txn
```

Export

```
$SQOOP_HOME/bin/sqoop export --connect <jdbc-url> -table <table-
name> --hcatalog-table txn
```

6. Installing and Configuring Flume in HDP

This document describes the process of manually installing and configuring Apache Flume for use with the Hortonworks Data Platform (HDP). Flume is the 1.x release line of the Flume project.

Use the following links to install and configure Flume for HDP

- Understand Flume
- Install Flume
- Configure Flume
- Start Flume
- HDP and Flume
- A Simple Example

What follows is a very high level description of the mechanism. For much greater detail, see the Flume html documentation set that is installed with Flume. Once you have installed Flume, the documentation set can be accessed at `file:///usr/lib/flume/docs/index.html`. The *"Flume User Guide"* is available at `file:///usr/lib/flume/docs/FlumeUserGuide.html`. The same documentation is also available at the Flume website, flume.apache.org.

6.1. Understand Flume

Flume is a top level project at the Apache Software Foundation. While it can function as a general-purpose event queue manager, in the context of Hadoop it is most often used as a log aggregator, collecting log data from many diverse sources and moving them to a centralized data store.

Note

What follows is a very high level description of the mechanism. For much greater detail, see the Flume html documentation set that is installed with Flume. Once you have installed Flume, the documentation set can be accessed at `file:///usr/lib/flume/docs/index.html`. The *"Flume User Guide"* is available at `file:///usr/lib/flume/docs/FlumeUserGuide.html`.

6.1.1. Flume Components

A Flume data flow is made up of five main components: Events, Sources, Channels, Sinks, and Agents.

6.1.1.1. Events

An event is the basic unit of data that is moved using Flume. It is similar to a message in JMS and is generally small. It is made up of headers and a byte-array body.

6.1.1.2. Sources

The source receives the event from some external entity and stores it in a channel. The source must understand the type of event that is sent to it: an Avro event requires an Avro source.

6.1.1.3. Channels

A channel is an internal passive store with certain specific characteristics. An in-memory channel, for example, can move events very quickly, but does not provide persistence. A file based channel provides persistence. A source stores an event in the channel where it stays until it is consumed by a sink. This temporary storage allows source and sink to run asynchronously.

6.1.1.4. Sinks

The sink removes the event from the channel and forwards it on either to a destination, like HDFS, or to another agent/dataflow. The sink must output an event that is appropriate to the destination.

6.1.1.5. Agents

An agent is the container for a Flume data flow. It is any physical JVM running Flume. The same agent can run multiple sources, sinks, and channels. A particular data flow path is set up through the configuration process.

6.2. Install Flume

Flume is included in the HDP repository, but it is not installed automatically as part of the standard HDP installation process.

6.2.1. Prerequisites

1. You must have installed at least core Hadoop on your system using one of the available methods. See HDP Deployment Options for more information on your options.
2. You must have set up your JAVA_HOME environment variable per your operating system. See the various installation guides above for more information.

6.2.2. Installation

To install Flume, from a terminal window type:

[For RHEL or CentOS]

```
yum install flume
yum install flume-node
```

or

[For SLES]

```
zypper install flume
zypper install flume-node
```

6.2.3. Users

The installation process automatically sets up the appropriate `flume` user and `flume` group in the operating system.

6.2.4. Directories

The main Flume files are located in `/usr/lib/flume` and the main configuration files are located in `/etc/flume/conf`.

6.3. Configure Flume

You configure Flume by using a properties file, which is specified on Flume start-up. There is a template for this file in the configuration directory: `/etc/flume/conf/flume-conf.properties.template`.

Note

There is also a file `flume-env.sh.template` in the `conf` directory. This file can be used to set environment variables automatically at start-up.

6.4. Start Flume

There are two options for starting Flume.

- Start Flume directly. On the Flume host:

  ```
  /etc/rc.d/init.d/flume-node start
  ```

- Start Flume as a service. On the Flume host:

  ```
  service flume-node start
  ```

6.5. HDP and Flume

Flume ships with many source, channel, and sink types. For use with HDP the following types have been thoroughly tested:

6.5.1. Sources

- Exec (basic, restart)

- Syslogtcp
- Syslogudp

6.5.2. Channels

- Memory
- File

6.5.3. Sinks

- HDFS: secure, nonsecure
- HBase

6.6. A Simple Example

The following snippet shows some of the kinds of properties that can be set using the properties file. For more detailed information, see the *"Flume User Guide"*.

```
agent.sources = pstream
agent.channels = memoryChannel
agent.channels.memoryChannel.type = memory
agent.sources.pstream.channels = memoryChannel
agent.sources.pstream.type = exec
agent.sources.pstream.command = python print_stream.py 200 10
agent.sinks = hdfsSink
agent.sinks.hdfsSink.type = hdfs
agent.sinks.hdfsSink.channel = memoryChannel
agent.sinks.hdfsSink.hdfs.path = hdfs://hdp/user/root/flumetest
agent.sinks.hdfsSink.hdfs.fileType = SequenceFile
agent.sinks.hdfsSink.hdfs.writeFormat = Text
```

The source here is defined as an `exec` source, which simply means that the agent runs a given command on start-up which streams data to `stdout`, where the source gets it. In this case, the command is a Python script written for test purposes. The channel is defined as an in-memory channel and the sink is an HDFS sink.

7. Using Cascading

Cascading is an application framework for Java developers to simply develop robust Data Analytics and Data Management applications on Apache Hadoop.

Cascading v2.1 is certified with HDP v1.3 and later. Cascading can be downloaded from here.

For more information on using Cascading, see Cascading User Guide.