Hortonworks Data Platform

Data Integration Services with HDP

(Aug 1, 2014)

docs.hortonworks.com

Hortonworks Data Platform: Data Integration Services with HDP

Copyright © 2012-2014 Hortonworks, Inc. Some rights reserved.

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 YARN, 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.

Except where otherwise noted, this document is licensed under
Creative Commons Attribution ShareAlike 3.0 License.
http://creativecommons.org/licenses/by-sa/3.0/legalcode

Table of Contents

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 systems, and use Hadoop to power large-scale data analysis across the enterprise.

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

- Enables users to read/write from/to Hadoop as a data source/sink.
- HCatalog Metadata services enables users to import raw data into Hadoop (HBase and HDFS), and to create and manage schemas.
- Pig and Hive are used to analyze these data sets.
- Enables 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 a 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 on 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, execute the following command on the gateway machine as the administrator user (HDFS user) to create a home directory:

  ```
  % hadoop dfs -mkdir /user/hdptestuser
  ```
- Ensure that the user launching the TOS has appropriate permissions on the HDP cluster.

EXAMPLE: If `hdptestuser` is responsible for launching TOS, 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.

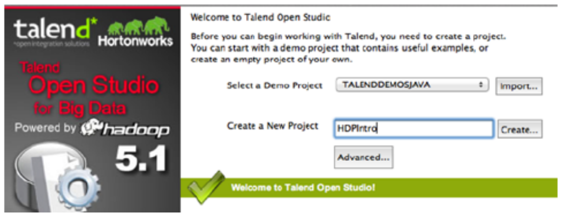
   b. After the download is complete, unzip the contents in an install location.

   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), then click the **Create** button.



Welcome to Talend Open Studic

Before you can begin working with Talend, you need to create a project.
You can start with a demo project that contains useful examples, or create an empty project of your own.

Select a Demo Project: TALENDDEMOSJAVA

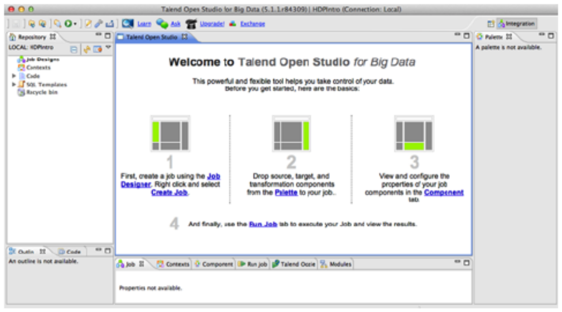
Create a New Project: HDPIntro

   b. Click **Finish** on the **New Project** dialog.

   c. Select the newly created project, then 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. The Talend Open Studio (TOS) main window appears and is now ready for use.



Welcome to Talend Open Studio *for Big Data*

This powerful and flexible tool helps you take control of your data.
Before you get started, here are the basics:

1 First, create a job using the Job Designer. Right click and select Create Job.

2 Drop source, target, and transformation components from the Palette to your job..

3 View and configure the properties of your job components in the Component tab.

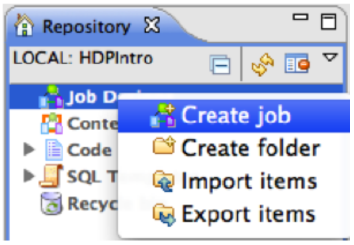
4 And finally, use the Run Job tab to execute your Job and view the results.

1.2.2. Writing a Talend Job for Data Import

This section describes how to design a simple job for importing a file into the Hadoop cluster.

1. Create a new job.

   a. In the Repository tree view, right-click the **Job Designs** node, then select **Create job**.



   b. In the New Job, wizard provide a name (for example HDPJob), then click **Finish**.

   c. An empty design workspace corresponding to the Job name opens.

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
```

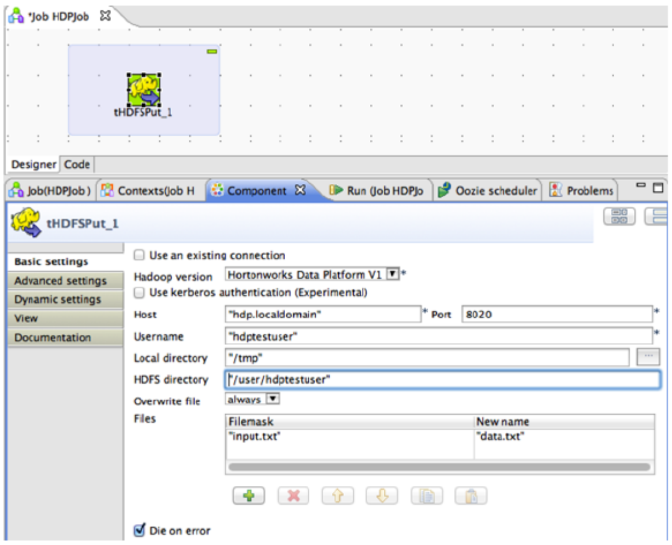
3. Build the job. Jobs are composed of components that are available in the Palette.

   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 the component in its **Basic Settings** view.

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



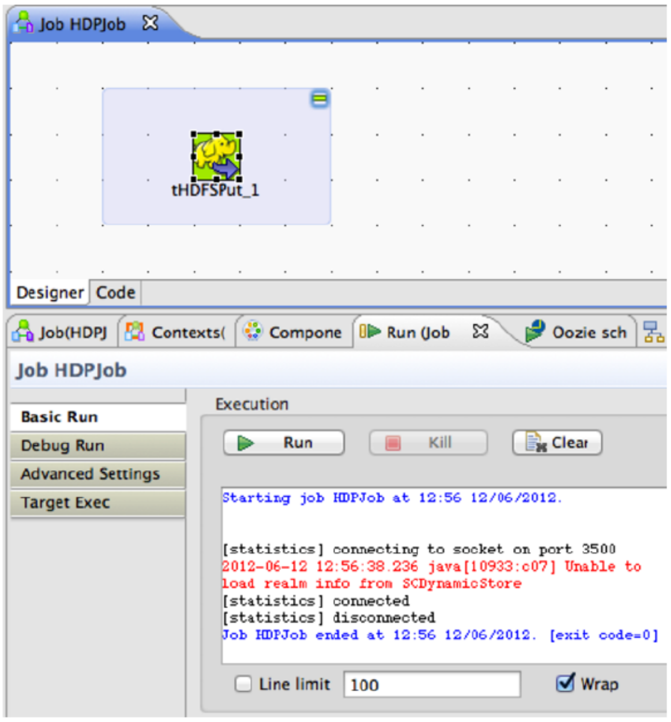
☐ Use an existing connection
Hadoop version: Hortonworks Data Platform V1
☐ Use kerberos authentication (Experimental)
Host: "hdp.localdomain" Port: 8020
Username: "hdptestuser"
Local directory: "/tmp"
HDFS directory: "/user/hdptestuser"
Overwrite file: always
Files:

Filemask	New name
"input.txt"	"data.txt"

☑ Die on error

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:



```
Starting job HDPJob at 12:56 12/06/2012.

[statistics] connecting to socket on port 3500
2012-06-12 12:56:38.236 java[10933:c07] Unable to
load realm info from SCDynamicStore
[statistics] connected
[statistics] disconnected
Job HDPJob ended at 12:56 12/06/2012. [exit code=0]
```

☐ Line limit 100 ☑ Wrap

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 the Pig component from the Big Data Palette.

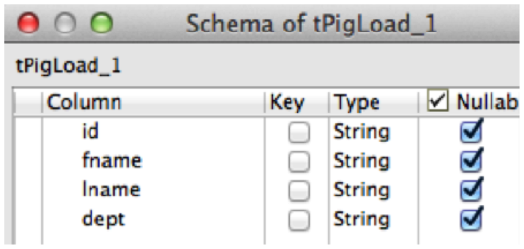
   a. Expand the **Pig** tab in the Big Data Palette.

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

2. Define basic settings for the Pig component.

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

   b. Click the **Edit Schema** button (“**...**” button). Define the schema of the input data as shown below, then click **OK**:

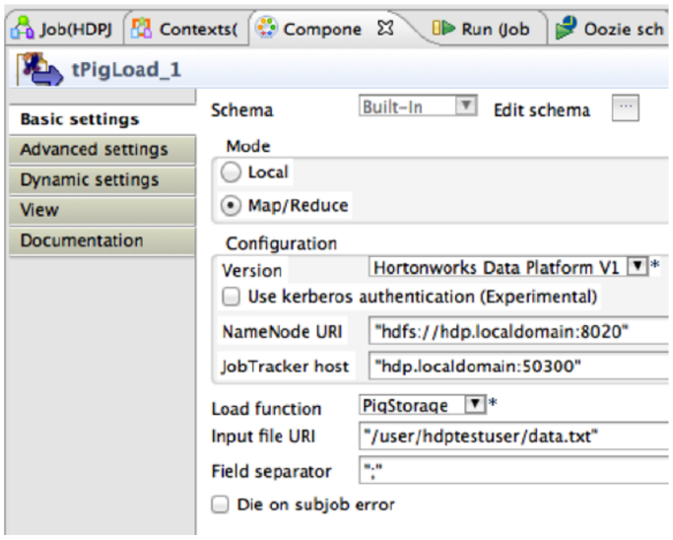


Column	Key	Type	✓ Nullab
id	☐	String	✓
fname	☐	String	✓
lname	☐	String	✓
dept	☐	String	✓

   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 hosts correspond to accurate values available in your HDP cluster. The Input File URI corresponds to the path of the previously imported `input.txt` file.



Schema: Built-In — Edit schema

Mode: ( ) Local (•) Map/Reduce

Configuration

Version: Hortonworks Data Platform V1

☐ Use kerberos authentication (Experimental)

NameNode URI: "hdfs://hdp.localdomain:8020"

JobTracker host: "hdp.localdomain:50300"

Load function: PigStorage

Input file URI: "/user/hdptestuser/data.txt"

Field separator: ";"

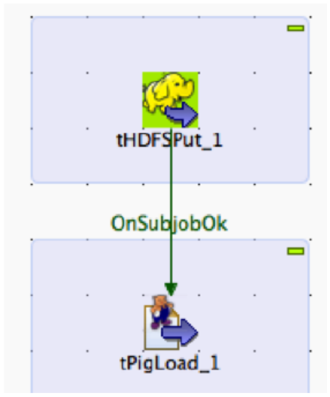
☐ Die on subjob error

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**).

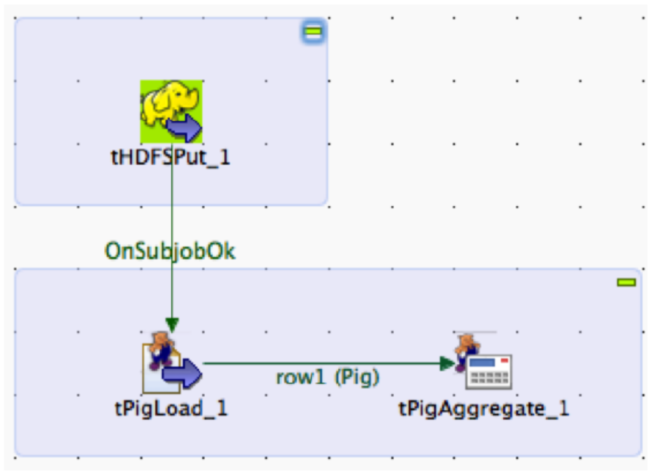


4. Add and connect Pig aggregate component.

   a. Add the component **tPigAggregate** next to **tPigLoad**.

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

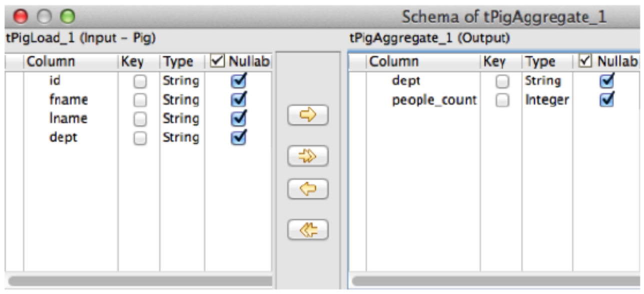
   c. Click on **tPigAggregate**.



5. Define basic settings for the 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:



Schema of tPigAggregate_1

tPigLoad_1 (Input – Pig)

Column	Key	Type	Nullable
id	☐	String	✓
fname	☐	String	✓
lname	☐	String	✓
dept	☐	String	✓

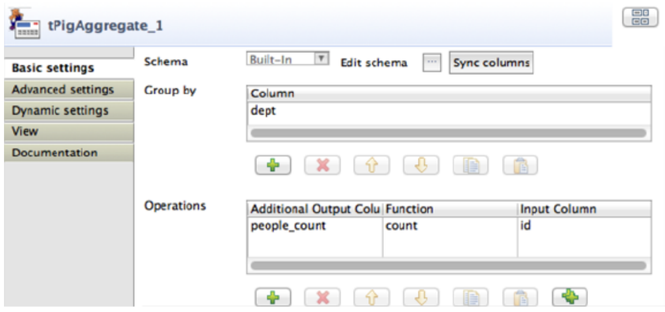
tPigAggregate_1 (Output)

Column	Key	Type	Nullable
dept	☐	String	✓
people_count	☐	Integer	✓

6. Define aggregation function for the data.

   a. For `Group by` add a Column and select `dept`.

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



Schema: Built-In Edit schema

Group by

Column
dept

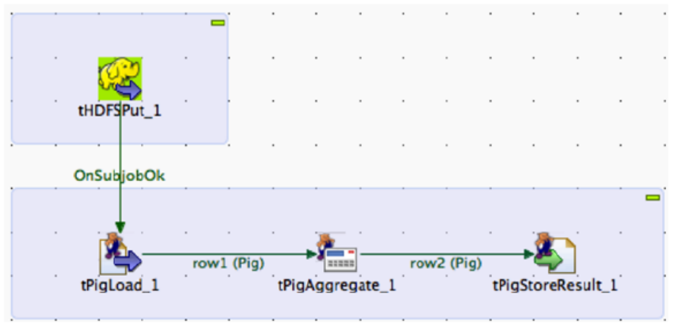
Operations

Additional Output Colu	Function	Input Column
people_count	count	id

7. Add and connect Pig data storage component

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

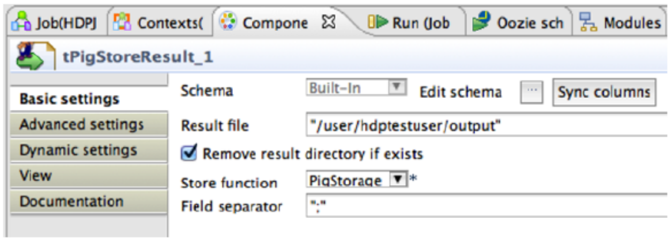
b. From the contextual menu, right-click on `tPigLoad`, select Row -> Pig Combine and click on `tPigStoreResult`.



8. Define basic settings for the 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:



Schema: Built-In
Result file: "/user/hdptestuser/output"
✓ Remove result directory if exists
Store function: PigStorage
Field separator: ";"

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 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 defines a simple SQL query language, called QL, that enables users familiar with SQL to query the data. At the same time, this language also allows programmers who are familiar with the MapReduce framework to be able 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.

In this document:

- Hive Documentation
- New Feature: Vectorization
- Hive ODBC Driver
- Hive Security without Grant/Revoke
- Hive Troubleshooting
- Hive JIRAs

2.1. Hive Documentation

Documentation for Hive can be found in wiki docs and javadocs.

1. Javadocs describe the Hive API.
2. The Hive wiki is organized in four major sections:
    - General Information about Hive
        - Getting Started
        - Presentations and Papers about Hive
        - Hive Mailing Lists
    - User Documentation
        - Hive Tutorial
        - HiveQL Language Manual
        - Hive Operators and Functions
        - Hive Web Interface

    - Hive Client
    - HiveServer2 Client
    - Avro SerDe
- Administrator Documentation
    - Installing Hive
    - Configuring Hive
    - Setting Up the Metastore
    - Setting Up Hive Web Interface
    - Setting Up Hive Server
    - Hive on Amazon Web Services
    - Hive on Amazon Elastic MapReduce
- Resources for Contributors
    - Hive Developer FAQ
    - How to Contribute
    - Hive Developer Guide
    - Plugin Developer Kit
    - Unit Test Parallel Execution
    - Hive Architecture Overview
    - Hive Design Docs
    - Full-Text Search over All Hive Resources
    - Project Bylaws

2.2. New Feature: Vectorization

Vectorization allows Hive to process a batch of rows together instead of processing one row at a time. Each batch consists of a *column vector* which is usually an array of primitive types. Operations are performed on the entire column vector, which improves the instruction pipelines and cache usage. HIVE-4160 has the design document for vectorization and tracks the implementation of many subtasks.

2.2.1. Enable Vectorization in Hive

To enable vectorization, set this configuration parameter:

- `hive.vectorized.execution.enabled=true`

When vectorization is enabled, Hive examines the query and the data to determine whether vectorization can be supported. If it cannot be supported, Hive will execute the query with vectorization turned off.

2.2.2. Log Information about Vectorized Execution of Queries

The Hive client will log, at the `info` level, whether a query's execution is being vectorized. More detailed logs are printed at the `debug` level.

The client logs can also be configured to show up on the console.

2.2.3. Supported Functionality

The current implementation supports only single table read-only queries. DDL queries or DML queries are not supported.

The supported operators are selection, filter and group by.

Partitioned tables are supported.

These data types are supported:

- tinyint
- smallint
- int
- bigint
- boolean
- float
- double
- timestamp
- string

These expressions are supported:

- Comparison: >, >=, <, <= , =, !=
- Arithmetic: plus, minus, multiply, divide, modulo
- Logical: AND, OR
- Aggregates: sum, avg, count, min, max

Only the ORC file format is supported in the current implementation.

2.2.4. Unsupported Functionality

All datatypes, file formats, and functionality *not listed in the previous section* are currently unsupported.

Two unsupported features of particular interest are the logical expression NOT and the `cast` operator. For example, a query such as `select x,y from T where a = b` will not vectorize if a is integer and b is double. Although both int and double are supported, casting of one to another is not supported.

2.3. 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 from here.
- The instructions on installing and using this driver are available here.

2.4. Hive Security without Grant/Revoke

By default, Hive in HDP uses HCatalog's storage-based authorization model instead of the basic Hive authorization model. Storage-based authorization manages access with file permissions, and the HiveQL statements GRANT and REVOKE have no effect. See Metastore Server Security in Hive Authorization and the HCatalog document Storage Based Authorization for more information.

2.5. Troubleshooting Hive

MySQL is the default database used by the Hive metastore. Depending on several factors, such as the version and configuration of MySQL, Hive developers may encounter an error message similar to the following:

```
An exception was thrown while adding/validating classes) : Specified key was
 too long; max key length is 767 bytes
```

Administrators can resolve this issue by altering the Hive metastore database to use the Latin1 character set, as shown in the following example:

```
mysql> ALTER DATABASE <metastore_database_name> character set latin1;
```

2.6. Hive JIRAs

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

3. 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.

Start the HCatalog CLI with the command '`<hadoop-install-dir>\hcatalog-0.5.0\bin\hcat.cmd`'.

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 next 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

3.1. 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

3.2. Using WebHCat

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

Note

WebHCat was originally named *Templeton*, and both terms may still be used interchangeably. For backward compatibility the Templeton 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

3.2.1. Technical Update: WebHCat Standard Parameters

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

3.2.1.1. Security

The current version supports two types of security:

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

3.2.1.1.1. 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'
```

3.2.1.1.2. 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."
}
```

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. Migrating HBase Data from HDP 1.3.x to HDP 2.0.x

This section describes how to manually migrate data between two HBase clusters, where the source cluster runs HBase 0.94 and the destination cluster runs HBase 0.96 and is empty of all data. This data migration may occur as part of a larger platform migration from HDP 1.3.x to HDP 2.0.x. HBase administrators can choose between two upgrade paths, depending on whether the organization can tolerate downtime for the source HBase cluster. Both scenarios assume that the HBase administrator has previously set up the destination HBase cluster with version 0.96.

Upgrading HBase with Significant Downtime

Use the following manual upgrade procedure for HBase if your organization can tolerate downtime for the entire HBase cluster.

1. Shut down both the source and destination HBase clusters.
2. Rename the HBase root folder in HDFS on the destination cluster. For example, rename `/apps/hbase` to `/apps/hbase_OLD`.
3. Remove any ZooKeeper data on the destination cluster.
4. Use the **hadoop distcp** command to move all data files from the HBase root directory in HDFS of the source cluster to the HBase root directory in the destination cluster.

   The command leaves the migrated data files using the 0.94 layout.
5. Run the **upgrade** command on the destination HBase cluster as the HBase user to convert the data files to the 0.96 layout:

   ```
   sudo su -l $HBASE_USER -c "hbase upgrade -execute"
   ```

   You should see a completed Znode upgrade with no errors.
6. Start the HBase services on the destination cluster as the HBase user:

   ```
   sudo su -l $HBASE_USER -c "/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/
   hbase/conf start master"
   sudo su -l $HBASE_USER -c "/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/
   hbase/conf start regionserver"
   ```
7. Check processes.

   ```
   ps -ef | grep -i master
   ps -ef | grep -i region
   ```

Upgrading HBase with Minimal Downtime

Use the following procedure to upgrade HBase if your organization requires minimal downtime for the source HBase cluster.

1. Shut down the destination HBase cluster.

2. Note the current time stamp.

   ```
   date +%s
   ```

3. Rename the HBase root folder in HDFS on the destination cluster. For example, rename `/apps/hbase` to `/apps/hbase_OLD`.

4. Remove any ZooKeeper data on the destination cluster.

5. Use the **hadoop distcp** command to move all data files from the HBase root directory in HDFS of the source cluster to the HBase root directory in the destination cluster.

   The command leaves the migrated data files using the 0.94 layout.

6. Run the **upgrade** command on the destination HBase cluster as the HBase user to convert the data files to the 0.96 layout:

   ```
   sudo su -l $HBASE_USER -c "hbase upgrade -execute"
   ```

   You should see a completed Znode upgrade with no errors.

7. Start the HBase services on the destination cluster as the HBase user:

   ```
   sudo su -l $HBASE_USER -c "/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/
   hbase/conf start master"
   sudo su -l $HBASE_USER -c "/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/
   hbase/conf start regionserver"
   ```

8. Stop any applications from writing data to HBase.

9. Pass the time stamp noted in step 2 as an argument to the HBase **EXPORT** command on the source HBase cluster to migrate any data written since the time stamp.

10. Use the HBase **IMPORT** command to get exported tail data into the destination cluster.

11. Route traffic to the destination cluster.