HPCC Systems™

HDFS to HPCC Connector User's Guide

Boca Raton Documentation Team

LexisNexis® Risk Solutions

HDFS to HPCC Connector User's Guide

Boca Raton Documentation Team

We welcome your comments and feedback about this document via email to <docfeedback@hpccsystems.com> Please include **Documentation Feedback** in the subject line and reference the document name, page numbers, and current Version Number in the text of the message.

2013 Version 1.4.0.1

HDFS to HPCC Connector

Introduction

The HDFS to HPCC Connector (H2H Connector) provides a means to import data from Hadoop's HDFS into an HPCC Systems platform. It also supports exporting the data back to HDFS or exporting and merging it. This allows you to use an HPCC cluster in conjunction with your Hadoop-based cluster.

There are two varieties of the H2H Connector, one based on **libhdfs** and one based on **webhdfs**.

The H2H Connector is an add-on to an HPCC Cluster and consists of server-side components and ECL Macros that invoke them.

- **Server-side components:**
  - The executable (/opt/HPCCSystems/bin/libhdfsconnector or /opt/HPCCSystems/bin/webhdfsconnector)
  - The shell script (/opt/HPCCSystems/bin/hdfspipe)
  - The configuration file (/etc/HPCCSystems/hdfsconnector.conf)
- **ECL Macros (HDFSConnector.ecl)**
  - HDFSConnector.PipeIn

    Imports data from Hadoop's file system (HDFS) at an ECL dataset.
  - HDFSConnector.PipeOut

    Exports data from a Thor Cluster to Hadoop's file system (HDFS).
  - HDFSConnector.PipeOutAndMerge

    Exports data from a Thor Cluster to Hadoop's file system (HDFS) and merges the data.

    Not supported when using a webhdfs implementation.
- **The HDFS to HPCC Connector User's Guide**

Implementation Varieties

There are two implementations of the H2H connector available, one based on **libhdfs** and one based on **webhdfs**. Each one has a separate installation package. Before installing you should consider which variation is more appropriate for you.

Note: The libhdfs variety of the H2H Connector assumes that you installed Hadoop using a standard installation package, either Hadoop 1.x or later using the rpm or deb package. If you have installed Hadoop manually there may be some additional configuration required.

Comparison Table

libhdfs	webhdfs
Utilizes native HDFS API	Does not need local installation of Hadoop nor JVM
Requires local installation of Hadoop	Requires webhdfs be enabled on target HDFS system
Local JVM required	Target HDFS datanode hostnames must resolve locally
libhdfs.so required	PipeOutAndMerge is not supported, user responsible for merging file parts on Hadoop
	libcurl required

Installation and Configuration

Installing the Server-side components

The installation and package that you download is different depending on the operating system you plan to use.

Download the appropriate package for your operating system.

You must install the package on every HPCC node.

To install the package, follow the installation instructions for your operating system:

Centos/Red Hat/SuSe

- Install RPM with the -Uvh switch.

  This is the upgrade command and will perform an automatic upgrade if a previous version is installed or it will just install fresh if no other version has been installed.

```
sudo rpm -Uvh <rpm file name>
```

Note: For ANY version of SuSe you must set a password for the **hpcc** user on all nodes. One way to do do this is to issue the following command:

```
sudo passwd hpcc
```

Be sure to set the password on ALL nodes.

Ubuntu/Debian

For Ubuntu installations a Debian package (dpkg) is provided.

- To install the package, use:

```
sudo dpkg -i <deb filename>
```

Editing and distributing the Configuration file

Once you install the H2H Connector package, if you edit the configuration file, you must then push it out to all nodes.

1. SSH to a node where the package has been installed.
2. Edit the configuration file **:**

```
/etc/HPCCSystems/hdfsconnector.conf
```

The configuration file defines where Hadoop resources are located. Be sure these variables are all pointing to the correct locations. There are some configuration examples in the default configuration file.

Note: If you get a runtime error, such as "library can't be found," check the library paths in your configurations file and edit, if needed. The installed configuration file contains samples to help you.

The HDFS Connector writes log files to a folder named **mydataconnectors** in the the HPCC log directory (the HPCC log location can be set using Configuration Manager).

The default location is:

```
/var/log/HPCCSystems/mydataconnectors/
```

The log files are written following the following pattern:

```
libhdfsconnector.<nodeid>.<PID>.log
```

Or if you chose the webhdfs implementation:

```
webhdfsconnector.<nodeid>.<PID>.log
```

3. If you modify the default configuration file, you must copy the configuration file to all nodes.

   You can use a script (such as the *hpcc-push.sh* script) to make this a little easier.

```
sudo -u hpcc /opt/HPCCSystems/sbin/hpcc-push.sh \
            /etc/HPCCSystems/hdfsconnector.conf \
            /etc/HPCCSystems/hdfsconnector.conf
```

Installing the H2H ECL library to your ECL IDE source folder

The HDFS to HPCC Connector (H2H) library is a single ECL file containing three MACROs. These steps explain how to install to your ECL source repository.

1. Download the **HDFS Connector Library for IDE (ZIP)** file from the HPCC web portal: http://hpccsystems.com/products-and-services/products/modules/hadoop-integration

2. Extract the contents of the zip file to the ECL IDE source folder. Make sure to select the option to use the folder names from the Zip file.

   The ECL Source folder is typically located at **C:\Users\Public\Documents\HPCC Systems\ECL\My Files**.

   To locate your source folder, refer to your ECL IDE preferences, located at **Preferences** >> **Compiler** >> **ECL Folders**.

   When you are finished, the library will be in a repository folder named **DataConnectors**. It will contain one file named *HDFSConnector.ecl*, which contains the ECL Macros used to interface with the H2H Connector.

ECL Macros

The ECL Macros are provided as an Application Programming Interface (API) to the HDFS Connector.

HDFSConnector.PipeIn

HDFSConnector.PipeIn *(ECL_RS, HadoopFileName, Layout, HadoopFileFormat, HDFSHost, HDFSPort)*

ECL_RS	The ECL recordset definition name to generate.
HadoopFileName	The Hadoop data file name as it exists in the HDFS.
Layout	The ECL RECORD structure representing the structure of the Hadoop data.
HadoopFileFormat	The Hadoop data file format : FLAT | CSV[csvoptions]
csvoptions	[[,]SEPARATOR(f_delimiters)] [[,]TERMINATOR(r_delimiters)] [[,]QUOTE(characters)]
	default values for SEPARATOR: ',' TERMINATOR: '\n' QUOTE: '"
HDFSHost	The Hadoop DFS host name or IP address.
HDFSPort	The Hadoop NameNode port number.

The **HDFSConnector.PipeIn** macro is called to pipe in data from the Hadoop file system (HDFS) to a Thor Cluster.

Example:

```
#OPTION('pickBestEngine', 0);
IMPORT std;
IMPORT DataConnectors;
Layout_Flat := RECORD
  STRING10  fname;
  STRING10  lname;
  UNSIGNED1 prange;
  STRING10  street;
  UNSIGNED1 zips;
  UNSIGNED1 age;
  STRING2   birth_state;
  STRING3   birth_month;
  UNSIGNED1 one;
  UNSIGNED8 id;
END;
DataConnectors.HDFSConnector.PipeIn(MyDataFile,
                          '/user/hadoop/test/MyData1',
                          Layout_Flat, FLAT,
                          '192.168.56.120',
                          54310);
OUTPUT(MyDataFile);
```

```
#OPTION('pickBestEngine', 0);
IMPORT std;
IMPORT DataConnectors;
Layout_CSV := RECORD
  STRING10 fname;
  STRING10 lname;
  STRING3  prange;
  STRING10 street;
  STRING5  zips;
  STRING3  age;
  STRING2  birth_state;
  STRING3  birth_month;
  STRING3  one;
  STRING20 id;
END;
DataConnectors.HDFSConnector.PipeIn(MyDataFile,
                           '/user/Administrator/test/MyData1',
                           Layout_CSV, CSV(SEPARATOR('|')),
                           '192.168.56.120',
                           54310);
OUTPUT(MyDataFile);
```

HDFSConnector.PipeOut

HDFSConnector.PipeOut *(ECL_RS, HadoopFileName, Layout, HadoopFileFormat, HDFSHost, HDFSPort, HDFSUser)*

ECL_RS	The ECL recordset to export.
HadoopFileName	The fully qualified target HDFS file name.
Layout	The structure which describes the ECL_RS recordset.
HadoopFileFormat	The Hadoop data file format : FLAT | CSV[csvoptions]
csvoptions	[[,]SEPARATOR(f_delimiters)] [[,]TERMINATOR(r_delimiters)] [[,]QUOTE(characters)]
	default values for SEPARATOR: ',' TERMINATOR: '\n' QUOTE: '"
HDFSHost	The Hadoop DFS host name or IP address.
HDFSPort	The Hadoop NameNode port number.
HDFSUser	HDFS username to use to login to HDFS in order to write the file. This user must have permission to write to the target HDFS location.

The **HDFSConnector.Pipeout** macro writes the given *ECL_RS* recordset to the target HDFS system in file parts -- one file part for each HPCC Thor node. These parts are written to the directory:

```
/user/local/hadooop/HadoopFileName-parts/
```

Where the *HadoopFileName* directory will take the name specified. You can then use other means to merge the file parts or you can use **HDFSConnector.PipeOutAndMerge** to do both tasks.

Examples:

```
#OPTION('pickBestEngine', 0);
IMPORT std;
IMPORT DataConnectors;
Layout_Flat :=RECORD
  STRING10 fname;
  STRING10 lname;
  UNSIGNED1 prange;
  STRING10 street;
  UNSIGNED1 zips;
  UNSIGNED1 age;
  STRING2 birth_state;
  STRING3 birth_month;
  UNSIGNED1 one;
  UNSIGNED8 id;
END;
MyDataFile := DATASET('~certification::full_test_distributed',Layout_Flat, FLAT);
//piping out hpcccertrecords to a flat file in HDFS called /user/hadoop/test/cert1,
DataConnectors.HDFSConnector.PipeOut(MyDataFile,
                            '/user/hadoop/test/MyData1',
                            Layout_Flat, FLAT,
                            '192.168.56.120',
                            54310
                            'hadoopusername' );
```

```
#OPTION('pickBestEngine', 0);
IMPORT std;
IMPORT DataConnectors;
Layout_CSV := RECORD
  STRING10 fname;
  STRING10 lname;
  STRING3  prange;
  STRING10 street;
  STRING5  zips;
  STRING3  age;
  STRING2  birth_state;
  STRING3  birth_month;
  STRING3  one;
  STRING20 id;
END;
MyDataFile := DATASET('~certification::full_test_distributed',Layout_CSV, CSV);
//piping out hpcccertrecords to a CSV file in HDFS called /user/hadoop/test/cert1,
DataConnectors.HDFSConnector.PipeOut(MyDataFile,
                            '/user/hadoop/test/MyData1',
                            Layout_CSV, CSV,
                            '192.168.56.120',
                            54310
                            'hadoopusername' );
```

HDFSConnector.PipeOutAndMerge

HDFSConnector.PipeOutAndMerge *(ECL_RS, HadoopFileName, Layout, HadoopFileFormat, HDFSHost, HDFSPort, HDFSUser)*

ECL_RS	The ECL recordset to export.
HadoopFileName	The fully qualified target HDFS file name.
Layout	The structure which describes the ECL_RS recordset.
HadoopFileFormat	The Hadoop data file format : FLAT | CSV[csvoptions]
csvoptions	[[,]SEPARATOR(f_delimiters)] [[,]TERMINATOR(r_delimiters)] [[,]QUOTE(characters)]
	default values for SEPARATOR: ',' TERMINATOR: '\n' QUOTE: '"
HDFSHost	The Hadoop DFS host name or IP address.
HDFSPort	The Hadoop NameNode port number.
HDFSUser	HDFS username to use to login to HDFS in order to write the file. This user must have permission to write to the target HDFS location.

The **HDFSConnector.PipeOutAndMerge** macro writes the given *ECL_RS* recordset to the target HDFS system in file parts and merges them together to form a single target file on the HDFS system.

NOTE: PipeOutAndMerge is NOT supported in the webhdfs implementation of HDFS Connector.

Example:

```
#OPTION('pickBestEngine', 0);
IMPORT std;
IMPORT DataConnectors;
Layout_Flat :=RECORD
  STRING10  fname;
  STRING10  lname;
  UNSIGNED1 prange;
  STRING10  street;
  UNSIGNED1 zips;
  UNSIGNED1 age;
  STRING2   birth_state;
  STRING3   birth_month;
  UNSIGNED1 one;
  UNSIGNED8 id;
END;
MyDataFile := DATASET('~certification::full_test_distributed',Layout_Flat, FLAT);
DataConnectors.HDFSConnector.PipeOutAndMerge(MyDataFile,
                                    '/user/hadoop/test/MyData1',
                                    Layout_Flat, FLAT,
                                    '192.168.56.120',
                                    54310,
                                    'hadoopusername' );
```

```
#OPTION('pickBestEngine', 0);
IMPORT std;
IMPORT DataConnectors;
Layout_CSV := RECORD
  STRING10 fname;
  STRING10 lname;
  STRING3  prange;
  STRING10 street;
  STRING5  zips;
  STRING3  age;
  STRING2  birth_state;
  STRING3  birth_month;
  STRING3  one;
  STRING20 id;
END;
MyDataFile := DATASET('~certification::full_test_distributed',Layout_CSV, CSV);
DataConnectors.HDFSConnector.PipeOutAndMerge(MyDataFile,
                                    '/user/hadoop/test/MyData1',
                                    Layout_CSV, CSV,
                                    '192.168.56.120',
                                    54310,
                                    'hadoopusername' );
```