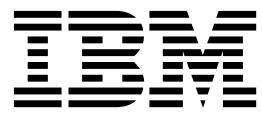


IBM Fluid Query  
Release 1.7.1

*User Guide*



**Note**

Before using this information and the product it supports, read the information in “Notices” on page A-1

**Revised: February 24, 2017**

This edition applies to IBM Fluid Query release 1.7.1 and to all subsequent releases until otherwise indicated in new editions.

© **Copyright IBM Corporation 2016, 2017.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

## About this publication . . . . . v

Audience . . . . . v

If you need help . . . . . v

How to send your comments . . . . . v

## Chapter 1. Getting started . . . . . 1-1

Product overview . . . . . 1-1

What's new in version 1.7.1 . . . . . 1-1

Best practices . . . . . 1-2

    Security and authentication . . . . . 1-2

    Data connector best practices . . . . . 1-5

    Data movement best practices . . . . . 1-8

    Best practices for complex scenarios . . . . . 1-11

Tutorials . . . . . 1-15

## Chapter 2. IBM Fluid Query installation 2-1

Software prerequisites . . . . . 2-1

    Supported Hadoop environments . . . . . 2-2

    Supported database environments . . . . . 2-2

Installing IBM Netezza Analytics . . . . . 2-2

Installing IBM Fluid Query on NPS . . . . . 2-4

    Upgrading IBM Fluid Query . . . . . 2-5

    Removing the data connector software . . . . . 2-7

Installing or upgrading the data movement feature 2-8

    Uninstalling the data movement feature from

    Hadoop . . . . . 2-10

## Chapter 3. Data connector . . . . . 3-1

Configuring a connection for predefined connectors 3-1

    JDBC drivers for predefined data connectors . . . 3-2

    Data type mapping for predefined connectors . . . 3-7

    Kerberos configuration file . . . . . 3-9

Configuring a connection to any JDBC data source 3-10

    Configuring a connection with hardcoded

    values . . . . . 3-11

    Configuring a connection with variables . . . 3-11

    Configuring a connection with customized data

    types mapping. . . . . 3-13

    Configuring a connection with Kerberos

    authentication . . . . . 3-16

    Generic connector sample configurations . . . 3-16

Encrypting password in SQL connector

    configuration file . . . . . 3-18

Registering the data connector functions . . . 3-19

    About local and remote mode functions . . . 3-21

    Assigning privileges to run the data connector

    functions . . . . . 3-27

    Revoking privileges to run the data connector

    functions . . . . . 3-28

Using data connectors . . . . . 3-30

    Running SQL queries using the data connector 3-30

    Using views to simplify user queries . . . 3-31

Unregistering the data connector functions . . . 3-32

## Chapter 4. Data movement . . . . . 4-1

Data movement algorithm . . . . . 4-2

Support for Hadoop-specific file formats . . . . 4-3

Checksum functionality for data movement . . . 4-3

Preparing configuration XML files . . . . . 4-4

    Data import configuration XML . . . . . 4-5

    Data export configuration XML . . . . . 4-14

    Preparing the configuration XML file for the

    Hadoop file formats . . . . . 4-20

    Append modes for import and export . . . . 4-23

    Using checksum to check data consistency . . 4-25

    Encrypting the configuration XML files . . . 4-27

Configuring and running data movement. . . . 4-27

    Validating data movement configuration . . . 4-28

    Running data movement from Hadoop . . . . 4-28

    Running data movement from NPS systems . . 4-32

    Running data movement from other systems . 4-34

Importing or exporting multiple tables at a time 4-35

Importing data to Hadoop with views created on

NPS . . . . . 4-36

Importing data to partitioned and clustered Hive

tables. . . . . 4-37

Exporting data from Hadoop . . . . . 4-39

    Exporting previously imported files . . . . . 4-39

    Exporting text files from Hadoop . . . . . 4-40

    Exporting a Hive table that was not imported

    from NPS . . . . . 4-41

Querying compressed NZBAK files from Hadoop 4-43

    Querying compressed NZBAK files on

    BigInsights 3 . . . . . 4-43

    Querying compressed NZBAK files on

    BigInsights 4 . . . . . 4-43

    Querying compressed NZBAK files on

    Hortonworks . . . . . 4-44

    Querying compressed NZBAK files on

    Cloudera . . . . . 4-44

    Querying compressed NZBAK files on BigSQL 4-45

Data type mapping for data movement . . . . . 4-46

## Chapter 5. Troubleshooting . . . . . 5-1

Resolving problems after downgrading IBM Fluid

Query to version 1.5 . . . . . 5-1

Data connector troubleshooting . . . . . 5-1

    Data connector limitations . . . . . 5-2

    Data connector known issues . . . . . 5-2

Data movement troubleshooting . . . . . 5-3

    Data movement known issues and limitations 5-8

    Troubleshooting import and export of Hadoop

    formats . . . . . 5-12

Troubleshooting connection problems . . . . . 5-15

Troubleshooting problems with Kerberos . . . . 5-16

IBM Fluid Query log files . . . . . 5-18

    Data connector log files. . . . . 5-18

    Data movement log files . . . . . 5-19

    Working with log collector. . . . . 5-19

<b>Chapter 6. Command and function reference</b>	<b>6-1</b>
The fqConfigure script	6-1
The fqDownload script	6-6
The fqRegister script	6-8
The fqRemote script	6-11
The fqAdmin script	6-12
The FqRead function	6-13
The <b>toHadoop</b> and <b>fromHadoop</b> functions	6-14
IBM support scripts	6-16
The fqLogCollector script	6-16
The fqCheckConfiguration script	6-16
The fqCheckJDBCconnect script	6-19

<b>Chapter 7. Appendix: Using IBM Fluid Query with MapR</b>	<b>7-1</b>
Configuring a connector to MapR	7-1
Running data movement between NPS and MapR	7-2
<b>Notices</b>	<b>A-1</b>
Trademarks	A-3
<b>Accessibility features for IBM Fluid Query</b>	<b>B-1</b>
<b>Index</b>	<b>X-1</b>

---

## About this publication

This document describes the IBM® Fluid Query feature for the IBM Netezza® platform. The guide describes how to install, configure, and use the data connector and data movement capabilities for querying and accessing data stored in Hadoop service providers and other supported data repositories.

---

## Audience

You should be familiar with the basic operation and concepts of the IBM Netezza system. You should also be familiar with Java style function declarations, NPS® database commands, and system administration. To complete some of the procedures described in the later sections, you must be able to log in to the Netezza appliance as the nz user to run operating system level commands and scripts. SQL query tasks require you to connect to Netezza databases as database user accounts to run SQL commands.

---

## If you need help

If you are having trouble using the IBM Netezza appliance, follow these steps:

1. Try the action again, carefully following the instructions for that task in the documentation.
2. Go to the IBM Support Portal at: <http://www.ibm.com/support>. Log in using your IBM ID and password. You can search the Support Portal for solutions. To submit a support request, click the **Service Requests & PMRs** tab.
3. If you have an active service contract maintenance agreement with IBM, you can contact customer support teams by telephone. For individual countries, visit the Technical Support section of the IBM Directory of worldwide contacts (<http://www.ibm.com/support/customer/care/sas/f/handbook/contacts.html>).

---

## How to send your comments

You are encouraged to send any questions, comments, or suggestions about the IBM Netezza documentation. Send an email to [netezza-doc@wwpdl.vnet.ibm.com](mailto:netezza-doc@wwpdl.vnet.ibm.com) and include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

We appreciate your suggestions.



---

## Chapter 1. Getting started

Read this section to learn about IBM Fluid Query, new features introduced in version 1.7.1, best practices, and tutorials.

---

### Product overview

IBM Fluid Query helps to address Big Data requirements around access to a wide range of structured databases existing across an enterprise. Federation technology enables the ability to query remote database objects across IBM on-premises and Cloud architectures, as well as third party commercial, open source and Hadoop distributions. The product also delivers a native bulk data copy capability that enables PureData System for Analytics to move or copy data in bulk with Hadoop distributions.

IBM Fluid Query allows queries to be sent to other database sources for processing in the Cloud, on-premise, Hadoop, as well as distributed and software defined environments. You can quickly query more data for discovery and deeper insights within systems across your data ecosystem.

It also allows rapid transfer of database level data between Hadoop and IBM PureData System for Analytics. It extends IBM PureData System for Analytics by allowing colder data to be offloaded to a lower cost-per-terabyte Hadoop environment.

---

### What's new in version 1.7.1

IBM Fluid Query version 1.7.1 offers a set of new features and improvements.

- You can now automatically create views on NPS of the data that you import to Hadoop. The functionality is called automatic federation. For more information, see “Importing data to Hadoop with views created on NPS” on page 4-36.
- Hive partitioning and clustering is now supported when moving data from NPS. For more information, see “Importing data to partitioned and clustered Hive tables” on page 4-37.
- You can now export a Hive table that was not previously imported from NPS. For more details, see “Exporting a Hive table that was not imported from NPS” on page 4-41.
- You can now use checksum calculation to verify data integrity and consistency after the data movement process. For more information, see “Checksum functionality for data movement” on page 4-3.
- The process of data movement from Hadoop was improved with the `fdm.sh` script. You no longer have to use `nzcodec.jar` and export the Hadoop classpath.
- The so called “local mode” of data movement, when IBM Fluid Query connects to Hive directly in order to create table metadata, is now deprecated. Remote mode is used instead by default, that is IBM Fluid Query uses a JDBC driver to connect to Hive or BigSQL. This approach reduces a number of limitations.
- Security was enhanced with the ability to encrypt passwords that are stored in connector configuration files using the 128-bit AES key. For more information see “Encrypting password in SQL connector configuration file” on page 3-18.

- The installation procedure was modified: You are asked to provide the paths to Hadoop /lib and /conf files during installation, to reduce additional configuration and simplify productive use of fast data movement.
- IBM Fluid Query has been certified for the following distributions: Big SQL 4.2, Hortonworks 2.5, Cloudera Impala 5.9, Spark 1.6.1 and DB2 LUE v11.1.

---

## Best practices

Read this section to learn about best practices, recommendations, and useful tips when using IBM Fluid Query.

### General considerations

- Use the latest available version of IBM Fluid Query. You can download it at IBM Fix Central. Type **IBM Netezza Applications** in the **Product selector** field and select **FLUIDQUERY\_1.7**.
- Read the latest version of documentation. It is available online in IBM Knowledge Center. Open **PureSystems > PureData Systems > PureData Systems for Analytics** and select the version that you want to use.

## Security and authentication

Read about best practices for password security, user rights and authentication with Kerberos.

### Passwords encryption

It is advisable to encrypt user passwords that are stored in configuration XML files.

- Encrypting password in SQL connector configuration file:

The password is encrypted by default using the static encryption key. With additional parameters used when running `fqConfigure.sh` you can use the automatically generated 128-bit AES key, or generate your own key. For more information see “Encrypting password in SQL connector configuration file” on page 3-18.

- Encrypting password in data movement configuration files:

You can encrypt the import or export configuration XML files by using the `nzcodec.jar` file with specific command arguments. Detailed instruction is available at “Encrypting the configuration XML files” on page 4-27.

- Encrypting password for generic connector:

When defining configuration for a generic connector, you can hardcode database name and user password values in the XML file, but they are stored in plain text.

If you do not specify password in the XML file, password prompt will be used instead when running the `fqConfigure` script.

### Authenticating with Kerberos

Depending on your environment configuration, and which IBM Fluid Query features you use, Kerberos configuration steps may differ.

**Important:** This section is applicable only if Kerberos authentication is already configured in your Hadoop environment.

### Configuring Kerberos authentication for predefined connectors

Following is an example procedure for configuring a connector for Impala with Kerberos.



Assumptions:

- Impala JDBC drivers are already copied to `/nz/export/ae/products/fluidquery/libs/cloudera/impala`
- User must be able to execute `kinit` from NPS machine. This step is performed internally by the `fqConfigure` script.
- Scripts `fqConfigure.sh` and `fqRegister.sh` must be run as user `nz`.

Configuration steps:

1. Configure connection to Impala using service principal `sr3dusr@internal.domain.com`:

```
/nz/export/ae/products/fluidquery/fqConfigure.sh --host cdts1hdpdn04d.rxcorp.com
--port 21050 --service-principal impala/cdts1hdpdn04d.rxcorp.com@INTERNAL.DOMAIN.COM
--client-principal sr3dusr/cdts1hdpdn04d.rxcorp.com@internal.domain.com
--krb5-conf /tmp/krb5.conf --config myConfig --provider cloudera --service impala
```

Principal password prompt is presented.

**Tip:**

- Alternatively you can use keytab: `--keytab path_to_keytab`
  - The following part of the command is optional: `--krb5-conf /tmp/krb5.conf`. If you do not specify it, then a default file `/etc/krb5.conf` is used. If you use the default file, ensure that it contains accurate information. For more information see “Kerberos configuration file” on page 3-9.
2. Register functions in NPS database:

```
/nz/export/ae/products/fluidquery/fqRegister --config myConfig --db myDb
```

**Tip:** Optionally, you can add the `--udtf myFuncName` parameter to specify custom name to the `fqRead` function.

3. Query the database using the registered function, for example:

```
nzsql myDb
select * from table with final (fqRead('impalaSchema','impalaTable'));
```

**Note:**

- All queries on the Impala end are executed as `sr3dusr@internal.domain.com` that was provided as an argument to `fqConfigure.sh`.
- The NPS user needs to have permission to execute Fluid Query functions.

## Configuring generic connector with Kerberos authentication

The procedure and examples are described in “Configuring a connection with Kerberos authentication” on page 3-16

## Fast data movement with Kerberos authentication

If you want to perform import and export operations from Hadoop using Kerberos authentication, follow the configuration steps described in “Configuring connection from Hadoop to NPS with Kerberos authentication” on page 4-29.

## Troubleshooting

Read about limitations and common errors when using Kerberos in “Troubleshooting problems with Kerberos” on page 5-16.

### Database user credentials

When configuring connectors, you must specify database credentials to external data source as well as to Netezza system. Read this section to understand how to manage user credentials.

To set up a connector to the external database service provider you must 1) configure and 2) register a connection. These two steps are separate processes.

1. To configure a connection to your external database service provider, run the `fqConfigure` script. You specify parameters for the external data source, such as IP address, port, database name, user name and password. Information about the connector is then stored on a file system in a properties file.
2. To register a connector run the `fqRegister` script. You must specify the NPS-side parameters for the connection defined in the configuration step: NPS database where the connector functions will be created, user name and password for the NPS database, custom function name etc. This step registers an IBM Fluid Query function inside the Netezza database. The registered function will then be used directly in an SQL statement to run query on the external database. After this step, the registered function is connected with the configuration to the external data source.

You can register more than one function using the same connector configuration.

When you run the registered function to read data from a table that is part of the external database, you must be connected to the NPS database where the function has been registered. The executed function connects to the external database using credentials provided during configuration and stored in the properties file.

In a situation when you need multiple users to access multiple external databases, you must run a separate `fqConfigure` script for each database and each user, specifying a different configuration file name using the `--config file_name` parameter, so that the previous configuration is not overwritten. A new connection to the external database is then created, but it only differs with username and password. You must then register the functions with different names.

When using other database providers, you do not have to specify the database name for the `fqConfigure` script. In these cases, you only have to run the script once per user, and you can pass the database name as an argument.

### Example

The following example presents how user credentials must be configured to access different external databases.

Use case: `nps_user` for `nps_database` needs to access data in two external DB2 databases:

- `DB2_Sales`
- `DB2_Finance`

Follow these steps to configure connections:

1. Run the `fqConfigure` script twice:

- To connect to DB2\_Sales database with DB2\_Sales\_user who can access this database:  

```
[nz@nzhost-h1 ~]$ ./fqConfigure.sh --host 192.0.2.1
--port 50000 --database DB2_Sales --username DB2_Sales_user --config MyConfig_Sales
--provider ibm --service DB2
```

- To connect to the DB2\_Finance database with DB2\_Finance\_user who can access this database:

```
[nz@nzhost-h1 ~]$ ./fqConfigure.sh --host 192.0.2.1
--port 50000 --database DB2_Finance --username DB2_Finance_user
--config MyConfig_Finance --provider ibm --service DB2
```

Two connection configuration files are created as a result, MyConfig\_Sales.properties and MyConfig\_Finance.properties. You use these files in the next step.

2. Run the fqRegister script twice to register the functions in the nps\_database for the nps\_user, so that this user can access both DB2 databases using the configured connections. You need two functions to be registered with two different names. Use the --udtf option to specify function name, and the --config option to provide the name for the properties file that you created in the previous step.

- Register function fqRead\_DB2\_Sales:

```
[nz@nzhost-h1 ~]$ ./fqRegister.sh --user nps_user --db nps_database
--udtf fqRead_DB2_Sales --config MyConfig_Sales
```

- Register function fqRead\_DB2\_Finance:

```
[nz@nzhost-h1 ~]$ ./fqRegister.sh --user nps_user --db nps_database
--udtf fqRead_DB2_Finance --config MyConfig_Finance
```

**Tip:** If you need to query both the Finance and Sales database in one query, then the DB2 user that you specify for fqConfigure must have permissions for both databases.

## Data connector best practices

Read about best practices and optimizing performance when using data connectors.

### How to build views to achieve better performance using IBM Fluid Query

Using IBM Fluid Query you can create a view that points to the external data. A view is a database object that can be used to build data model. The views functionality has some limitations from the performance point of view, which you should be aware of.

In the following example, a list of employees is stored on dashDB in table employees. This list includes managers and engineers. IBM Fluid Query function called dashDB has been created to use data from the external data source on dashDB. A query that uses data from dashDB has been replaced by a view called dashDB\_employees.

```
create or replace view dashDB_employees as select * from table with
final(dashDB('','','select * from employees'));
```

To query the whole list, you can use the view without any limitations:

```
select * from dashDB_employees
```

To query only engineers, use the view with the following limitation:

```
select * from dashDB_employees where position = 'Engineers'
```

In this case, IBM Fluid Query sends a query that is part of the dashDB\_employees view to dashDB, and any limitations are performed on the NPS side. In other words, all of the records of table employees are sent to NPS, and then position is restricted by the Netezza box. This implementation has impact on performance. Firstly, the limitation tasks are performed on the Netezza box and most of them are handled only by the host. It is not effective, as Netezza is using FPGA and SPU. Secondly, loading this amount of data on Netezza may have impact on Netezza resources.

**Note:** The above scenario can be implemented when you do not expect much data coming from the employees table. For the huge amount of data that is part of the external table, the recommendations following this note have to be considered.

The solution that might improve performance, is to create two views, one for managers, one for engineers, and then use these views for your reports. In this way, you limit the amount of data that is loaded. Moreover, the work of filtering data is performed on the external data source, and not on NPS, which improves NPS performance.

Similarly, instead of the following view definition:

```
create or replace view dashDB_employees as select * from table with
    final(dashDB('','','select * from employees'));
```

use these:

```
create or replace view dashDB_Engineers as
select * from table with final(dashDB('','','select
    * from employees where position = 'Engineer')));
create or replace view dashDB_Managers as
select * from table with final(dashDB('','','select
    * from employees where position = 'Managers')));
```

Then, instead of the following query:

```
select count(*) from dashDB_employees where position = 'Engineer'
```

you can use this one:

```
select count(*) from dashDB_Engineers;
```

Starting with version 1.7.1, you can also automatically create views on NPS of the data that you import to Hadoop. The functionality is called automatic federation. For more information, see “Importing data to Hadoop with views created on NPS” on page 4-36.

## **How to build queries to improve performance by the use of NPS system capabilities with Fluid Query function**

Learn how to build queries involving a temporary table, which helps to achieve better execution time for SQL operations.

During join operation, data loaded from the external data source is processed using host memory, which affects performance. Using a temporary table allows to load data to NPS and then use hardware acceleration, which significantly improves the execution time of any SQL operations, especially a join operation.

In the following examples, there are two tables:

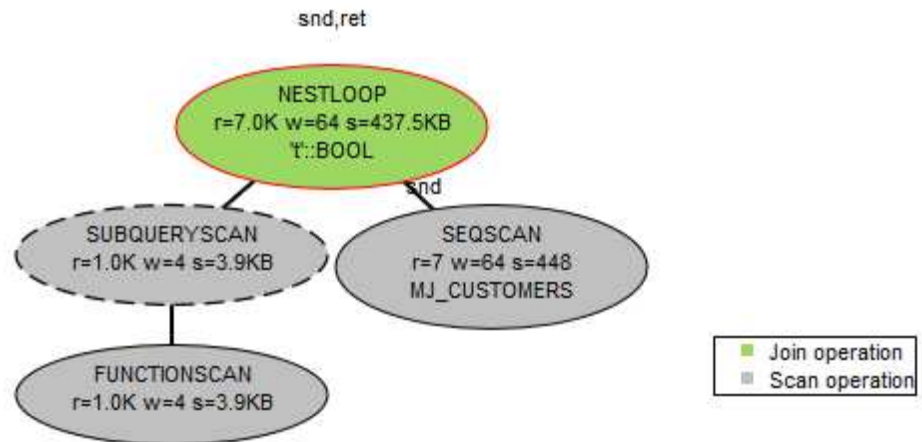
- mj\_customers - a view combined with a table from an external data source.
- mj\_orders - a view combined with a table from NPS.

Example 1 presents the use of a join operation of the two tables by id column. Graph 1 shows that the whole operation is performed on host.

Example 2 includes two steps. First, data from external table mj\_orders is loaded into a new table mj\_orders\_temp in NPS database. Then, the same join operation as in Example 1 is performed using the internal table. Graph 2 shows that most operations are performed on FPGA and SPU.

### Example 1

```
select name, age, address from mj_customers inner join mj_orders on mj_customers.id = id;
```



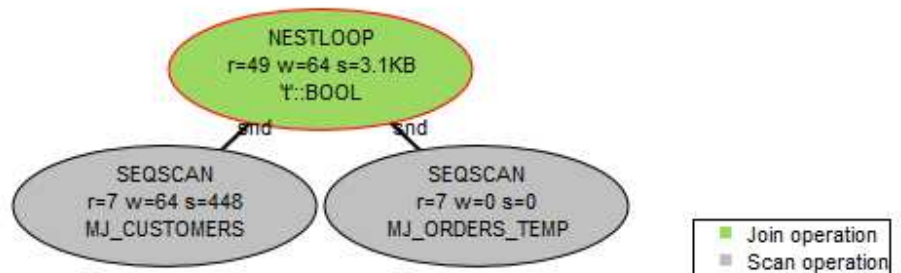
### Example 2

1. Create internal table on NPS as a result of query from external data source:

```
create table mj_orders_temp as select * from
mj_orders;
```

2. Run query using the internal table:

```
select name, age, address from mj_customers inner join mj_orders_temp on
mj_customers.id = id;
```



As shown in the examples, the use of temporary table might significantly improve the execution time of SQL operations. However, certain limitations apply: you must consider the time used to create the temporary table, and available storage on NPS.

## Using remote mode connection to optimize performance of SQL connector queries

When you register the data connector functions, you can specify whether the functions run in local or remote mode. Remote mode for connector is recommended.

In remote mode, the system runs only one query on the external data source, whereas two queries are run for local mode behavior, first to get table definition, and then the actual query. Hence, in local mode, there is a substantial overhead to get table definition, which is not present when running functions in remote mode.

Remote mode is especially useful when you plan to run a series of queries to the same tables, or run the same query multiple times.

To learn more about the differences between the local and remote mode, read “About local and remote mode functions” on page 3-21.

For remote mode (not local) , you can configure the number of concurrent queries and the number of queued queries in the configuration properties file. For more information see “Workload management considerations” on page 3-23

## Debugging data connector configuration

Read about log files when configuring data connectors.

After you install the data connector feature, you can obtain log files for various commands, tasks, and actions in the `/nz/export/ae/products/fluidquery/logs` directory.

If the `fqConfigure` command fails, there is a link to the log file in the error message.

When you solve the problems, you must run the `fqConfigure` script again.

Once you register the IBM Fluid Query functions in the NPS database with the `fqRegister` script, the software creates log files each time a query runs and calls the functions:

- When using local mode, two new log files are created every time you run a query.
- When using remote mode, query info is appended to a single file, which is created when you start the `fqRemote` service.

**Note:** It is important to clean up the `/nz/export/ae/products/fluidquery/logs/` log directory periodically. If remote mode of SQL connector is used, remote mode daemon must be stopped before the cleanup task, and restarted after the cleanup task.

## Data movement best practices

Read about best practices when running import and export operations with IBM Fluid Query.

### General considerations:

- The data movement feature transfers data directly between Hadoop data nodes and NPS. Therefore, a working network connection between all data nodes and NPS is a prerequisite.

- Run validation after you create configuration XML. “Validating data movement configuration” on page 4-28
- When you run data movement from NPS and encounter problems or errors, run the same operation from Hadoop as it provides more logs for troubleshooting.

## Data movement performance

To achieve high performance of the data movement functionality, check whether all the elements that contribute to the transfer are working properly.

Data movement involves several elements that contribute to the performance of the whole functionality. The data transfer is only as efficient as the weakest of those links. If you want to achieve the best transfer rates using IBM Fluid Query, check all the components that take part in the process of data movement:

1. NPS
2. Network
3. Fluid Query Hadoop job
4. HDFS
5. Hard drives

The overall speed depends equally on all these elements. Similarly to NPS, IBM Fluid Query uses parallel processing. Best results are achieved when the transfer is run in several streams. Due to some restrictions on NPS, the maximum number of streams cannot be higher than 30. Usually, the best performance can be achieved with 24 streams. The number of streams can be set in the XML configuration files with the **`nz.fq.splits`** parameter.

The following is a detailed description of the elements that are involved in data movement and the factors that can influence their performance:

**NPS** A properly configured instance of IBM Fluid Query can transfer data with the maximum speed provided by NPS. An important performance factor is data distribution. IBM Fluid Query retrieves data in parallel using several JDBC sessions at the same time. Data is split based on its distribution on data slices. If data is not equally distributed throughout all data slices or if there are significant differences in size, the largest data slices will require the longest time to unload their content. When the transfer starts, all streams are utilized equally, but at the end, some of them may already have finished the transfer, while others are still working. In such case, the average transfer speed is decreased. It is also important to monitor whether all unloads are started and that there are no waiting queries on the NPS scheduler.

### Network

This is a straightforward dependency as high transfer speed can only be achieved on fast network. For relatively small appliances, like N3001-002 or N3001-005, it is sufficient to use a 10Gb link. However, for more powerful appliances, like N3001-010 and higher, the recommended link is at least 20Gb. It is important to not only provide fast network but also to make it exclusively dedicated for IBM Fluid Query. All additional network activity may degrade the data movement speed.

### Fluid Query Hadoop job

There are two settings in IBM Fluid Query configuration that affect performance:

- **`nz.fq.splits`** - This setting defines how many parallel sessions to NPS are opened during import. If the number is too low, it may affect

performance as it also defines the number of hard drives which are used at the end. Usually the higher the better, however setting too many unloads may overload NPS.

- **nz.fq.compress** and **nz.fq.output.compressed** - These settings define how data is transferred from NPS to Hadoop. **nz.fq.compress** defines whether data is transferred over the network in plain text or in compressed binary. Compressed data is usually unloaded faster and has lower impact on the network but decompression extensively utilizes the Hadoop nodes. It is also important to monitor whether IBM Fluid Query mapper jobs are not restarted - one attempt only. Each failing attempt results in transfer speed degradation.

**HDFS** The Hadoop file system stores data on many nodes and many physical drives. Make sure that enough drives can be used during the transfer. During the import, if you have 24 splits configured, then 24 processes write data to hard drives. Each process usually writes to several drives at the same time, as HDFS keeps data in several files. This setting is defined by the **dfs.replication** parameter in `hdfs-site.xml`, by default it is 3. With such settings during import, you can expect 72 (3x24) drives used solely for Fluid Query import. If there are any jobs running at the same time, the number will be higher. On a system with fewer drives, some drives will be shared between several processes. In such case, the drive speed is limited drastically.

#### Hard drives

At the end of the transfer, all data is written to physical hard drives. Even though one file in HDFS can be located on many disks, at the moment of storing, data is written to one hard drive at the same time. In other words, one stream cannot perform faster than the speed of a drive. Problems appear when there are not enough drives and more than one process is written to the same disk.

### Enforcing direct data movement

With the **fq.data.directmovement** property you can influence the way in which data in Hadoop formats and in clustered or partitioned Hive tables is transferred.

By default, importing data to Hadoop formats or partitioned/clustered tables is a two-step process: first, data is sent to HDFS in text format, then it is converted to the selected format, and the text data is deleted. A possible limitation is that such process requires doubled free space on HDFS to store temporary files. Hence, to save the HDFS space, it is possible to write data directly to the selected format by setting the property `fq.data.directmovement=true`.

Note that when using such direct transfer, you save HDFS space, however the transfer itself is slower, especially for big files.

Moreover, when exporting files with `fq.data.directmovement=true`, the number of connections to NPS is not controlled by any means, which might cause performance issues.

The setting works for both import and export. The default value for this parameter is `false`.

For more information on the process of transferring data see “Data movement algorithm” on page 4-2.



## Best practices for complex scenarios

### How to build queries when using historical data

Learn about implementing IBM Fluid Query in a scenario where cold (historical) data is stored on Hadoop and current data is stored on NPS, and, periodically, data is moved from NPS to Hadoop. Review recommendations for query performance optimization.

### Example environment

The example scenario that is used in this topic includes Stores Weekly Sales data. There are 3 226 200 records in total.

Year	COUNT
2012	617000
2013	795200
2014	854800
2015	850000
2016	109200

COUNT
3226200

Historical data (from 2012 to 2015) is stored on Hadoop (BigInsights) in table STORES\_WEEKLY\_SALES in database MJ. Current data (2016) is stored on NPS in table stores\_weekly\_sales in database MJ.

### Recommendation 1:

You can create views to optimize SQL statement and create a data model. To manage the Stores Weekly Sales data, three views are created on NPS:

- stores\_weekly\_sales\_historical - to query historical data only:  
create or replace view stores\_weekly\_sales\_historical as  
select \* from table with final  
(mj\_BigInsight\_123('mj','','select \* from STORES\_WEEKLY\_SALES'));
- stores\_weekly\_sales\_current - to query current data only:  
create or replace view stores\_weekly\_sales\_current as  
select \* from table with final  
(mj\_BigInsight\_123('mj','','select \* from stores\_weekly\_sales'));
- stores\_weekly\_sales\_all - to query all data:  
create or replace view stores\_weekly\_sales\_all as  
select \* from stores\_weekly\_sales\_current  
union all  
select \* from stores\_weekly\_sales\_historical;

To query all data, regardless of whether the data is stored on NPS or Hadoop, you can use view stores\_weekly\_sales\_all.

```
select * from stores_weekly_sales_all;  
select count(*) from stores_weekly_sales_all where "Year" = 2013;  
select count(*) from stores_weekly_sales_all where "Year" = 2016;
```

To query only historical data, you can use view stores\_weekly\_sales\_historical.

```
select * from stores_weekly_sales_historical;  
select count(*) from stores_weekly_sales_historical where "Year" = 2013;
```

To query only current data you can use view stores\_weekly\_sales\_current.

```
select * from stores_weekly_sales_current;
select count(*) from stores_weekly_sales_current where "Year" = 2016;
```

### How to optimize performance when using views:

Two views that are used, `stores_weekly_sales_historical` and `stores_weekly_sales_all` are built based on the IBM Fluid Query function. It means that any query limitation performed on these two views is performed on NPS, not on Hadoop. One query is executed on Hadoop: the query that is part of the IBM Fluid Query function. To better understand it, let's take a look at the definition of view `stores_weekly_sales_historical` again:

```
create or replace view stores_weekly_sales_historical as
  select * from table with final
    (mj_BigInsight_123('mj','', 'select * from STORES_WEEKLY_SALES'));
```

In this example, query `select * from STORES_WEEKLY_SALES` is always sent to Hadoop using IBM Fluid Query function called `mj_BigInsight_123`. When running the following query:

```
select count(*) from stores_weekly_sales_all where "Year" = 2013
```

First, the data that is the result of query `select * from STORES_WEEKLY_SALES` executed on Hadoop is sent to NPS, and then the limitation where `"Year" = 2013` is performed on NPS. It may have impact on NPS performance.

### **Recommendation 2:**

It is a good solution to measure time that is required to finish a query using view and a query without view. You must consider that the size of cold data will grow, which may have impact on overall performance as well.

In the example environment the following results were achieved:

- Query with predefined view took 62 seconds:  

```
select count(*) from
  stores_weekly_sales_all where "Year" = 2013;
```
- When the view was replaced by direct query (hence limitation was performed on Hadoop), the query took 40 seconds:  

```
select count(*) from (
  select * from table with final (mj_BigInsight_123('mj','',
    'select * from STORES_WEEKLY_SALES where Year=''2013'''))
  union all
  select * from stores_weekly_sales where "Year" = 2013
) as t;
```

### **Recommendation 3:**

You can create more views based on your business use case. In the following example, the objective is to query data for specific year. The solution is to create one view for each specific year. In this way, initial limitation of data is performed on Hadoop:

```
create or replace view stores_weekly_sales_historical_2012 as
  select * from table with final (mj_BigInsight_123('mj','',
    'select * from STORES_WEEKLY_SALES where Year=''2012'''))
create or replace view stores_weekly_sales_historical_2013 as
  select * from table with final (mj_BigInsight_123('mj','',
    'select * from STORES_WEEKLY_SALES where Year=''2013'''))
```

```

create or replace view stores_weekly_sales_historical_2014 as
  select * from table with final (mj_BigInsight_123('mj','',
    'select * from STORES_WEEKLY_SALES where Year=''2014'''))
create or replace view stores_weekly_sales_historical_2015 as
  select * from table with final (mj_BigInsight_123('mj','',
    'select * from STORES_WEEKLY_SALES where Year=''2015'''))
create or replace view stores_weekly_sales_historical_2016 as
  select * from table with final (mj_BigInsight_123('mj','',
    'select * from STORES_WEEKLY_SALES where Year=''2016'''))

```

In the example environment, execution of the following query took 36 seconds:

```
select count(*) from stores_weekly_sales_historical_2013;
```

Count for year 2013 took 36 seconds (Note that the time is the same as in Recommendation 2):

```

select count(*) from (
  select * from stores_weekly_sales_historical_2013
  union all
  select * from stores_weekly_sales where "Year" = 2013
) as t;

```

#### **Recommendation 4:**

A stored procedure called STORES\_WEEKLY\_SALES\_HISTORICAL will provide the value of parameter Year into a query. The query is executed directly on Hadoop including any query limitation, and then the results are stored in a temporary table called stores\_weekly\_sales\_temp on NPS. General view stores\_weekly\_sales\_all will utilize the temporary table to query data from both NPS and Hadoop.

The flow is as follows:

1. Run stored procedure with parameter Year.
2. Stored procedure builds dynamic query with any limitations (in our example Year) and sends it to Hadoop.
3. Waiting for the results from Hadoop.
4. Results are stored in a temporary table on NPS.
5. Run general view to see all results.

View stores\_weekly\_sales\_historical is not required. It is replaced by the temporary table.

View stores\_weekly\_sales\_current (to query current data only) must not be changed:

```

create or replace view stores_weekly_sales_current as
  select * from stores_weekly_sales

```

General view stores\_weekly\_sales\_all (to query all data) must be changed by replacing stores\_weekly\_sales\_historical with the temporary table stores\_weekly\_sales\_temp:

```

create or replace view stores_weekly_sales_all as
  select * from stores_weekly_sales_current
  union all
  select * from stores_weekly_sales_temp ;

```

#### Definition of the procedure

Parameter Year is an example. You can create as many parameters as you want and pass them to the query.

```
CREATE OR REPLACE PROCEDURE STORES_WEEKLY_SALES_HISTORICAL(INTEGER)
RETURNS INTEGER
LANGUAGE NZPLSQL AS
BEGIN_PROC
DECLARE
    Year_Param ALIAS FOR $1;
    sql varchar(100);
BEGIN
    IF exists (select 1 from _v_table where TABLENAME = 'STORES_WEEKLY_SALES_TEMP') THEN
        drop table stores_weekly_sales_temp;
    END IF;
    sql := 'select * from STORES_WEEKLY_SALES where Year = ' || Year_Param;
    create table stores_weekly_sales_temp as
    (select * from table with final
    (mj_Cloudera_123('mj','', sql )));
END;
END_PROC;
```

### Running the stored procedure

To calculate a number of sales records from 2009 and current year run the following commands:

```
call stores_weekly_sales_historical(2009)
select count(*) from stores_weekly_sales_all;
```

### Definition of advanced procedure

Parameter Year is defined as a range.

```
CREATE OR REPLACE PROCEDURE STORES_WEEKLY_SALES_HISTORICAL(INTEGER, INTEGER)
RETURNS INTEGER
LANGUAGE NZPLSQL AS
BEGIN_PROC
DECLARE
    Year_From ALIAS FOR $1;
    Year_To ALIAS FOR $2;
    sql varchar(100);
BEGIN
    IF exists (select 1 from _v_table where TABLENAME = 'STORES_WEEKLY_SALES_TEMP') THEN
        drop table stores_weekly_sales_temp;
    END IF;
    sql := 'select * from STORES_WEEKLY_SALES where Year between ' || Year_From || ' and ' || Year_To;
    create table stores_weekly_sales_temp as
    (select * from table with final
    (mj_Cloudera_123('mj','', sql )));
END;
END_PROC;
```

### Running the stored procedure

To calculate a number of sales records between 2012 and 2013 and current year, run the following commands:

```
call stores_weekly_sales_historical(2012,2013)
select count(*) from stores_weekly_sales_all;
```

### Performance considerations

Performance is close to Recommendation 2. You must consider that the results of Hadoop query are loaded to NPS into a temporary table. This means that some disk resources are used. Nevertheless, union or join operations are performing faster because they are performed mostly on FPGA and SPU level. When you expect huge amount of data coming to NPS as a result of Hadoop query, this solution might work faster than the one in Recommendation 2.


## Summary

Depending on the business use case and the goal to be achieved, different options might be optimal:

- Recommendation 1: One global view allows you to use a static query, no matter where the data is stored (Hadoop or NPS).
- Recommendation 2: Dynamic query is performing faster, but every time a query has to be created from scratch, either using scripts, or other 3rd party applications, and it cannot be used as a data model.
- Recommendation 3: More predefined views allow you to use static query; from the performance point of view this solution is in the middle of Recommendations 1 and 2.
- Recommendation 4: When expecting huge amount of data coming to NPS as a result of Hadoop query, this solution might work faster than the one in Recommendation 2.

The following table presents query execution times in the described example environment. The query counted all records from year 2013.

*Table 1-1. Execution times for a query using different recommendations*

Recommendation	Duration in the example	Performance	Use as data model
1. Static, one view	62 seconds		YES
2. Dynamic query	40 seconds		NO
3. More views	36 seconds		YES
4. Stored procedure	<36 seconds		NO

## Notes on moving historical data

In the described example environment, every week an incremental import is performed. This is a two-steps process. First, you must import data to Hadoop in incremental mode, and then you must remove the already migrated records. Depending on the Hadoop and NPS configuration, the operations might be time-consuming. When data movement is in progress, any queries on data might provide wrong results. Therefore, any reports should be blocked for the time of data migration.

---

## Tutorials

Use tutorials available online to learn how to perform major tasks with IBM Fluid Query.

To access tutorials, open the online version of this documentation that is available at IBM Knowledge Center. Open **PureSystems > PureData Systems > PureData System for Analytics > IBM Fluid Query 1.7.0 > Getting started > Tutorials**.



---

## Chapter 2. IBM Fluid Query installation

The following sections describe software prerequisites and the installation of all the components of IBM Fluid Query.

IBM Fluid Query installation package contains the files that are required to install both, data connector and data movement features. Depending on which functionality you plan to use, you must install the package on NPS, on Hadoop, or on both. The installer automatically recognizes where you install the package.

---

### Software prerequisites

To use the data connection feature, you must have an NPS system, an environment to which you want to connect, such as Hadoop or DB2®, and the IBM Fluid Query software package.

The IBM Fluid Query software package is available from IBM Fix Central at <http://www.ibm.com/support/fixcentral>.

**Tip:** Use the following information to look up the package:

Product Group: Information Management  
Select from Information Management: IBM Netezza Applications  
Installed version: FLUIDQUERY 1.7  
Select the latest available fixpack from the list of fixes.

The following table shows the NPS system requirements and software revision levels.

*Table 2-1. NPS software requirements*

NPS release	Minimum IBM Netezza Analytics release	Recommended IBM Netezza Analytics release
7.0.2.x	2.5	2.5.4
7.0.4.x	2.5.4 (multiple schema support disabled)	3.0.1 (multiple schema support enabled)
7.1.0.x	3.0	3.0.2
7.2.0	3.0.2	3.2
7.2.0.2 and higher	3.0.2	3.2.1 and higher

**Note:**

- If you are planning to use Java 7 JDBC drivers in IBM Fluid Query, you must use IBM Netezza Analytics version 3.2.1 or higher. For more information on compatible JDBC drivers, refer to “JDBC drivers for predefined data connectors” on page 3-2.
- When using IBM Netezza Analytics version 3.2.1 and above, you cannot install or downgrade to IBM Fluid Query versions prior to 1.6, as these versions are incompatible. For more information see “Resolving problems after downgrading IBM Fluid Query to version 1.5” on page 5-1.

## Supported Hadoop environments

The IBM Fluid Query supports connections to the following Hadoop environments.

The IBM Fluid Query feature has been tested and verified to work with the following Hadoop service providers.

*Table 2-2. List of supported services and Hadoop environments*

Service	Provider		
	IBM BigInsights® versions 2.1, 3.0, 4, 4.1, 4.2	Cloudera versions 4.7, 5.3, 5.4, 5.5.1, 5.9	Hortonworks version 2.2, 2.3, 2.4, 2.5
Hive2	Yes	Yes	Yes
BigSQL	Yes	No	BigSQL 4.2
Impala	No	Yes	No

IBM Fluid Query supports JDBC-based connections to Hive2, BigSQL, and Impala services. You can connect to the Hadoop environment and run queries using SQL syntax. Only READ operations are supported.

## Supported database environments

Starting with IBM Fluid Query version 1.5, you can use the data connector feature to connect your NPS system to other database and data warehouse service providers in addition to the supported Hadoop service providers. Starting with version 1.6, the product features a generic connector, which allows you to connect IBM Fluid Query to any database provider with a JDBC driver.

IBM Fluid Query has been tested and verified to work with the following services.

*Table 2-3. List of supported database services*

Provider	Service	Version
IBM	IBM PureData® System for Analytics	
IBM	DB2	10.1 10.5
IBM	dashDB	N/A
IBM	PureData System for Operational Analytics	
Apache	Spark SQL	1.2.1 1.4
Oracle	Oracle Database	12.1.0.2
Generic	Generic	

---

## Installing IBM Netezza Analytics

The data connector feature requires the IBM Netezza Analytics software to be installed on the NPS appliance.



## Before you begin

If you have already installed IBM Netezza Analytics on your system, you can skip these steps. Follow this procedure only if you do not have IBM Netezza Analytics on your system, or if the IBM Fluid Query feature installation fails because it could not find the IBM Netezza Analytics software.

**Note:** IBM Netezza Analytics versions 3.2.1 and above are incompatible with IBM Fluid Query versions prior to 1.6.

## Procedure

1. To check whether IBM Netezza Analytics is installed and its version, run:

```
nzcm -inza
```

You can also check for IBM Netezza Analytics by running the following SQL command:

```
nzsql inza -c "select * from product"
```

2. Obtain the IBM Netezza Analytics software from IBM Fix Central at <http://www.ibm.com/support/fixcentral>.
3. Follow the documentation in the *IBM Netezza Analytics Administrator's Guide* to start the installation program. During the installation, you are prompted for a series of options. You must install the INZA packages and the INZA cartridge installer (shown below), but answer the other options as you prefer. Some suggested responses to the installation follow:

Install INZA packages? (y/n): **Enter y (required)**

Install Documentation packages? (y/n): **Enter n**

Please review the packages to install:

1) INZA package: YES

2) INZA Documentation package: NO

Do you wish to install the above selections?

Enter "y" to continue, "x" to exit

or any other key to be prompted to modify your selection: **Enter y**

Installing INZA packages...

Available zipped installation file(s):

[0] /export/home/nz/inza/v2.5.4/inza-2.5.4.zip

[1] A zipped file in a different directory or with a non-standard name.

Enter your selection: **Enter 0**

[OPTIONAL: Do you want to reset repository configuration? [y]/n **Enter y**

Would you like to run the INZA cartridge installer now? (y/n): **Enter y (required)**

Would you like to perform an Express (e) or Custom (c) install (e/c): **Enter c**

Install MapReduce components? (y/n): **Enter n**

Install Matrix components?

Note: Matrix components are also required for PCA, Kmeans, GLM and Linear Regression. (y/n): **Enter n**

Install IBM Netezza In-database Analytics components? (y/n): **Enter n**

Install Spatial components? (y/n): **Enter n**

Please review the components to install:

1) MapReduce: NO

- 2) Matrix: NO
  - 3) IBM Netezza In-database Analytics: NO
  - 4) Spatial: NO
- Do you wish install the above selections?  
Enter "y" to continue, "x" to exit  
or any other key to be prompted to modify your selection: **Enter y**
4. Restart your Bash session or run the following command:
- ```
source ~/.bashrc
```

---

## Installing IBM Fluid Query on NPS

Follow these steps to install the IBM Fluid Query software on the IBM PureData System for Analytics appliance. The same installation package is required for both, data connector and data movement features.

### Before you begin

Note that for IBM Fluid Query version 1.7, the installer requires that you have defined the following system variables on NPS:

- `NZ_USER=ADMIN`
- `NZ_PASSWORD` set to ADMIN's password

Password cached with the `nzpassword` command is not supported by the installer. If the variables are not set correctly, even though the installation is successful, you might get the following warning message when using IBM Fluid Query:

```
WARNING: Unable to register IFQAdmin. Try to execute
        "/nz/export/ae/products/fluidquery/fqRegister.sh --admin"
```

or

```
WARNING: Unable to register IFQAdmin. Try to execute
        "/nz/export/ae/products/fluidquery/fqRegister.sh --ifq-admin"
```

You must always use the `--ifq-admin` parameter, not `--admin`.

### Procedure

1. Download the `nz-fluidquery-version.tar.gz` file from IBM Fix Central at <http://www.ibm.com/support/fixcentral>, where *version* is a release version such as v1.7.1. Save the file in a location accessible to the Netezza active host.
2. Log in to the Netezza active host as the `nz` user.
3. Change to the directory where the `nz-fluidquery-version.tar.gz` file is located.
4. Decompress the file using the following command:  

```
gunzip nz-fluidquery-version.tar.gz
```
5. Extract the tar file using the following command:  

```
tar xvf nz-fluidquery-version.tar
```
6. As the **nz** user, run the installation command:  

```
./fluidquery_install.sh
```

Sample output follows:

```

-----
IBM Fluid Query Installer
(C) Copyright IBM Corp. 2015 All rights reserved.
Version 1.6.0.0 [Build 150908-161]
-----
Installing connector...
Checking if IBM Netezza Analytics is installed...
IBM Netezza Analytics is already installed. Ok.
Checking if IBM Netezza Analytics cartridge is installed...
IBM Netezza Analytics cartridge is already installed. Ok.
Checking for previous installation of IBM Fluid Query...
Previous installation of IBM Fluid Query does not exist in destination...
Extracting package to /nz/export/ae/products/fluidquery...
Copying connector files...
Copying license to /opt/nz/licenses/ifq...
Copying uninstaller to /nz/export/ae/products/fluidquery...
Installing Java...
Checking if backup of IBM Netezza Analytics Java exists...
Creating backup of IBM Netezza Analytics Java...
Backup of IBM Netezza Analytics Java created...
Installation successful! Package installed in /nz/export/ae/products/fluidquery
For details see the log file at /nz/var/log/fluidquery_install/
fluidquery_install.2015-05-28.10:37:43.log

```

## Results

After a successful installation, the installer automatically places the data connector setup files in the `/nz/export/ae/products/fluidquery/` directory.

**Note:** When you install the data connector bundle, the bundle also includes an **fqDownload.sh** command which is used for the data movement feature. The command is not used for the data connector support.

The installation requires the IBM Netezza Analytics software to be installed on the Netezza appliance. If the installer does not find a Netezza Analytics package, the script displays the following message:

IBM Netezza Analytics not found. Install IBM Netezza Analytics before installing the IBM Fluid Query software.

See “Installing IBM Netezza Analytics” on page 2-2 for more information. After you install IBM Netezza Analytics, run the **fluidquery\_install.sh** script again.

If you want to upgrade the data connector software, see “Upgrading IBM Fluid Query” for specific instructions.

## Upgrading IBM Fluid Query

If there is a new kit or fix pack available for IBM Fluid Query, follow these steps to upgrade the software on the IBM PureData System for Analytics appliance. The procedure is the same for both, data connector and data movement features.

### Before you begin

Note that for IBM Fluid Query version 1.7, the installer requires that you have defined the following system variables on NPS:

- `NZ_USER=ADMIN`
- `NZ_PASSWORD` set to ADMIN's password

Password cached with the `nzpassword` command is not supported by the installer. If the variables are not set correctly, even though the installation is successful, you might get the following warning message when using IBM Fluid Query:

```
WARNING: Unable to register IFQAdmin. Try to execute
        "/nz/export/ae/products/fluidquery/fqRegister.sh --admin"
```

or

```
WARNING: Unable to register IFQAdmin. Try to execute
        "/nz/export/ae/products/fluidquery/fqRegister.sh --ifq-admin"
```

You must always use the `--ifq-admin` parameter, not `--admin`.

## About this task

The procedure uses the same **fluidquery\_install.sh** script as the installation process. If the **fluidquery\_install.sh** detects that IBM Fluid Query is already installed on the system, the script displays a message prompting you to update the installed version of the software. If you answer **n** to the prompt, the upgrade exits and does not change the software.

## Procedure

1. Download the `nz-fluidquery-version.tar.gz` file from IBM Fix Central at <http://www.ibm.com/support/fixcentral>, where *version* is a release version, such as v1.7. Save the file in a location accessible to the Netezza active host.
2. Log in to the Netezza active host as the `nz` user.
3. Change to the directory where the `nz-fluidquery-version.tar.gz` file is located.
4. Decompress the file using the following command:  

```
gunzip nz-fluidquery-version.tar.gz
```
5. Extract the tar file using the following command:  

```
tar xvf nz-fluidquery-version.tar
```
6. Run the installation command to upgrade:  

```
./fluidquery_install.sh
```

Sample output follows. To upgrade, accept the **y** default shown below and press the *Enter* key.

```

-----
IBM Fluid Query Installer
(C) Copyright IBM Corp. 2015 All rights reserved.
Version 1.6.0.0 [Build 150908-161]
-----

Installing connector...
Checking if IBM Netezza Analytics is installed...
IBM Netezza Analytics is already installed. Ok.
Checking if IBM Netezza Analytics cartridge is installed...
IBM Netezza Analytics cartridge is already installed. Ok.
Checking for previous installation of IBM Fluid Query...
A previous installation of IBM Fluid Query was found.
Existing FluidQuery Version: 1.5.0.0 [Build 150630-186]
New FluidQuery Version: 1.6.0.0 [Build 150908-161]
Would you like to upgrade the existing IBM Fluid Query installation? (y/n) [y] Enter
Backing up existing installation files of IBM Fluid Query to
/nz/export/ae/products/fluidquery.backup.2015-09-09.09:22:43
Backup of the existing installation of IBM Fluid Query created...
Copying connector files...
Copying license to /opt/nz/licenses/ifq...
Copying uninstaller to /nz/export/ae/products/fluidquery...
Installing Java...
Checking if backup of IBM Netezza Analytics Java exists...
Backup of IBM Netezza Analytics Java already exists...
Installation successful! Package installed in /nz/export/ae/products/fluidquery
For details see the log file at /nz/var/log/fluidquery_install/
fluidquery_install.2015-09-09.09:22:32.log

```

## Results

After a successful installation, the installer automatically places the latest IBM Fluid Query files in the `/nz/export/ae/products/fluidquery/` directory, and creates the backup in the `/nz/export/ae/products/fluidquery.backup.date_id/` directory.

## Removing the data connector software

If you no longer want to use the data connector feature, follow this procedure to remove the functions and software from a IBM PureData System for Analytics appliance.

### About this task

To remove the data connector feature, you must unregister the data connector functions that were added to your databases, and you must remove the data connector scripts and configuration files from your system.

### Procedure

1. Log in to a database on the NPS system as the nz user.
2. Unregister the data connector functions using a command similar to the following. You must use a database user account such as admin or one that has privileges to drop functions. You might have to run this command multiple times if you or your users have registered the data connector functions in several databases or using different function names in the same database.

```
./fqRegister.sh --unregister [--udtf <func_name>] [--db <db_name>]
[--config <config_name>]
```

If you do not specify the `--udtf`, `--db`, or `--config` options, the command uses the function name defined in the `default.properties` configuration file and the `NZ_DATABASE` variable to identify the function names and database to search. You can specify the `--udtf` and `--db` or the `--config` file as needed to find the data connector functions.

3. Change to the `/nz/export/ae/products/fluidquery` and review the files stored there to see if you need any backups of the data connector files.
4. Navigate to the installation directory of IBM Fluid Query and run the `fluidquery_uninstall_connector.sh` script. The script prompts you before it deletes the configuration or library files, which you might want to do if you are planning to install the data connector again in the future. You can choose to uninstall all of the files, which will remove the data connector files from your NPS host.

---

## Installing or upgrading the data movement feature

If you plan to use the data movement functionality, you must first download the IBM Fluid Query package and install a set of files on your Hadoop nodes. When you run data movement operations from NPS, the package must also be installed on NPS.

The installation and the upgrade procedure of the data movement feature are identical.

### About this task

Installation of IBM Fluid Query on Hadoop is required when you plan to:

- Run fast data movement (import or export) from Hadoop
- Run fast data movement from NPS in mixed or binary mode
- Query compressed files from Hadoop

Depending on the mode that you will use when transferring the files and where you plan to run the movement operations from, you must install IBM Fluid Query on different nodes. The following scenarios are possible:

#### **Text mode of transfer (`nz.fq.compress=false` , `nz.fq.output.compressed=false`) and data movement initiated from NPS**

No installation on Hadoop is required.

#### **Mixed mode of transfer (`nz.fq.output.compressed=false`) and data movement initiated from NPS**

Installation on any single node is required. In fact, only the files installed on HDFS are used, so when installed on a single node they will be available to all other nodes.

#### **Text and mixed mode of transfer (`nz.fq.output.compressed=false`) and data movement initiated from Hadoop**

Installation on the node from which data transfer will be initiated is required.

#### **Compressed (binary) mode of transfer (`nz.fq.output.compressed=true`)**

Installation on all nodes where Hive services are running is required (`hiveserver2` and `hivemetastore`).

#### **Transfer to Hadoop specific formats**

Installation on all nodes where Hive services are running is required (`hiveserver2` and `hivemetastore`).

Note that in all cases, if data transfer is to be initiated from NPS, IBM Fluid Query must be installed on NPS. A working network connection between all data nodes and NPS is also a prerequisite in all cases.

**Note:** If Kerberos authentication is set up on Hadoop, you must run **kinit** before the installation to obtain the Kerberos ticket.

## Procedure

1. Download the `nz-fluidquery-version.tar.gz` package and save it in a location accessible to your Hadoop nodes.
2. Change to the directory where the `nz-fluidquery-version.tar.gz` file is located.
3. Run the following command to extract the `.tar.gz` package:

```
gunzip nz-fluidquery-version.tar.gz
```

4. Extract the `.tar` package:

```
tar -xvf nz-fluidquery-version.tar
```

5. As root, run the installation script:

```
./fluidquery_install.sh
```

If you are installing on Hadoop, you might be asked to provide paths to Hive or BigSQL `/lib` and `/conf` directories, if the paths on your system are other than default. These paths are then used when running the `fdm.sh` script to run data movement from Hadoop. The following table presents default paths depending on the service provider. If the installation script cannot locate the required files two times in a row, the installation process fails.

Table 2-4. Default paths for Hadoop files

| Service Provider            |        | /lib                                                                           | /conf                               |
|-----------------------------|--------|--------------------------------------------------------------------------------|-------------------------------------|
| BigInsights 3               | Hive   | /opt/ibm/biginsights/hive/lib/                                                 | /opt/ibm/biginsights/hive/conf/     |
|                             | BigSQL | /home/bigsql/sql/lib/java/                                                     | /opt/ibm/biginsights/hive/conf/     |
| BigInsights 4               | Hive   | /usr/iop/current/hive-client/lib/                                              | /usr/ibmpacks/bigsql/4.0/hive/conf/ |
|                             | BigSQL | /home/bigsql/sql/lib/java/                                                     | /usr/ibmpacks/bigsql/4.0/hive/conf/ |
| Hortonworks                 |        | /usr/hdp/current/hive-client/lib/                                              | /usr/hdp/current/hive-client/conf/  |
| Hortonworks 2.3.2 and above |        | /usr/hdp/current/atlas-server/hook/hive/;<br>/usr/hdp/current/hive-client/lib/ | /usr/hdp/current/hive-client/conf/  |
| Cloudera 4                  |        | /usr/lib/hive/lib/;<br>/usr/share/cmf/cloudera-navigator-server/libs/cdh4/     | /etc/hive/conf/                     |

Table 2-4. Default paths for Hadoop files (continued)

| Service Provider    | /lib                                            | /conf           |
|---------------------|-------------------------------------------------|-----------------|
| Cloudera 5          | /opt/cloudera/<br>parcels/CDH/lib/<br>hive/lib/ | /etc/hive/conf/ |
| Cloudera QuickStart | /usr/lib/hive/lib/                              | /etc/hive/conf/ |

## Results

The data movement files are now available on your Hadoop nodes.

**Note:** If you are installing data movement to the default directory, the installation logs are saved to the installation location. For default installations, the path is /fluidqueryLocal/var/log/fluidquery\_install. If you specify an alternate location using the `--datamove_path` option, the logs are saved in the /<datamove\_path>/var/log/fluidquery\_install directory.

## Uninstalling the data movement feature from Hadoop

Run the provided uninstallation script to uninstall the data movement feature.

### Procedure

1. Navigate to the installation folder of IBM Fluid Query on the service provider system.
2. As root, run the script `fluidquery_uninstall_datamove.sh`

## Results

The script automatically uninstalls the product.



---

## Chapter 3. Data connector

The IBM Fluid Query data connector feature allows you to access and query various data sources within your Big Data ecosystem from your IBM PureData System for Analytics appliances. In this version of the offering, you can query against structured data stored within various Hadoop service providers such as IBM BigInsights, Cloudera, and Hortonworks, and in other database environments such as DB2, dashDB™, other IBM PureData System for Analytics appliances, IBM PureData System for Operational Analytics, or Oracle.

To start using data connectors, you must perform the following steps, which are described in detail in further sections:

1. Depending on the service that you want to use, configure connections using *predefined connectors* or a *generic connector*.

The following data sources might be used with predefined connectors:

- IBM BigInsights
- Cloudera
- Hortonworks
- DB2
- dashDB
- IBM PureData System for Analytics
- IBM PureData System for Operational Analytics
- Oracle

You can also define a connection to any JDBC data source by using a generic connector feature. Various connection templates are available to facilitate the configuration process.

2. Register the functions in either local or remote mode.
3. Assign privileges to the NPS database users to run the functions.

You can then use the data connector user-defined table functions to create SQL queries that can select from structured data stored on these external file systems. You can combine the results from your Netezza Platform Software (NPS) database tables and the other data sources to create powerful combinations of analytics.

The following sections describe the steps to configure the connections, register the data connector functions, and use the functions in your SQL queries. There are also sections for workload management considerations when running these data connector queries on your system.

---

### Configuring a connection for predefined connectors

To use the data connector to query data on external environments, you must define a connection to each service that you plan to query and use. You can use a number of predefined connectors, which reduces configuration steps.

**Note:** Predefined connectors use default behavior for any parameters which are not specified in the JDBC URL. If you need to overwrite the defaults, you should use a generic connector instead as it allows for defining a number of parameters.

## Before you begin

You must install the data connector software and the JDBC drivers for your service providers before you can configure a connection. For more information see “JDBC drivers for predefined data connectors.”

### Procedure

1. Log in to the NPS active host as the nz user.
2. Change to the /nz/export/ae/products/fluidquery/ directory.
3. Run the **fqConfigure.sh** script to create a connection to a service provider. Sample commands follow.
  - To create a connection to Cloudera Impala and use Kerberos authentication:  

```
./fqConfigure.sh --host 192.0.2.1 --port 21050 --service-principal  
impala/myhost.example.com@example.com --client-principal myuser@example.com  
--krb5-conf /mypath/krb5.conf --config myConfig --provider cloudera  
--service impala
```
  - To create a connection to IBM BigInsights BigSQL and use a local user account and password:  

```
./fqConfigure.sh --host 192.0.2.2 --port 7052 --username biadmin  
--config biBigSql --provider ibm --service BIGSQL
```
  - To configure a connection to DB2:  

```
./fqConfigure.sh --host 192.0.2.2 --port 50000 --username admin  
--config myConfig --provider ibm --service db2 --database mydb
```
  - To configure a connection to PureData System for Operational Analytics:  

```
./fqConfigure.sh --host 192.0.2.2 --port 7052 --username admin  
--config myConfig --provider ibm --service pdoa --database mydb
```

For a list of options for the `./fqConfigure.sh` script, refer to the table “*The fqConfigure.sh input options*”.

### Results

If the connection is created successfully, the command displays the message:  
Connection configuration success.

If the command fails and displays an error message, there is a link to the log file in the message. Review the file, solve the problems, and run the `fqConfigure` script again.

For more information about the script and its options, see “The `fqConfigure` script” on page 6-1.

### What to do next

You can now register the data connector functions, as described in “Registering the data connector functions” on page 3-19.

## JDBC drivers for predefined data connectors

The data connector feature requires you to obtain JDBC drivers from your service providers and install them on the NPS appliance.

You only need to obtain drivers for the service provider that you are plan to use.

The following table lists the supported service providers, the location on the NPS appliance where the JDBC drivers must be stored, and the required JDBC driver

files. You only need to obtain drivers for the service provider that you are plan to use. The sample file list and the file revision numbers could change depending on the service provider and driver version that you use, as well as with updates to those software releases or JAR files.

The NPS path column shows the directory under the /nz/export/ae/products/fluidquery/ where you must save the JDBC files for each service provider that you use. Make sure that you store the JDBC files in the correct locations on the NPS appliance.

*Table 3-1. List of JDBC drivers for the supported service providers*

| Provider            | Service                                   | NPS path                                                                     | Sample file list                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------|-------------------------------------------|------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Oracle              | Oracle                                    | ./libs/oracle/oracle/                                                        | Drivers are available on the Oracle website.                                                                                                                                                                                                                                                                                                                                              |
| Apache              | Spark SQL                                 | ./libs/ibm/sparksql/<br>./libs/cloudera/sparksql/<br>./libs/horton/sparksql/ | As listed in this table for your provider's Hive.                                                                                                                                                                                                                                                                                                                                         |
| IBM                 | IBM PureData System for Analytics         | ./libs/ibm/pda/                                                              | nzjdbc3.jar                                                                                                                                                                                                                                                                                                                                                                               |
|                     | DB2                                       | ./libs/ibm/db2/                                                              | db2jcc.jar                                                                                                                                                                                                                                                                                                                                                                                |
|                     | dashDB                                    | ./libs/ibm/dashdb/                                                           | db2jcc.jar                                                                                                                                                                                                                                                                                                                                                                                |
|                     | PureData System for Operational Analytics | ./libs/ibm/pdoa/                                                             | db2jcc.jar                                                                                                                                                                                                                                                                                                                                                                                |
| IBM BigInsights 4.1 | Spark SQL                                 | ./libs/ibm/sparksql/                                                         | spark-assembly-1.4.1_IBM_2-hadoop2.7.1-IBM-8.jar<br>located in /usr/iop/current/spark-client/lib/                                                                                                                                                                                                                                                                                         |
| IBM BigInsights 4   | Hive 2                                    | ./libs/ibm/hive                                                              | hive-jdbc-0.14.0_IBM_8-standalone.jar<br>hadoop-common-2.6.0-IBM-7.jar<br>log4j.properties                                                                                                                                                                                                                                                                                                |
|                     | BigSQL                                    | ./libs/ibm/bigsql                                                            | db2jcc.jar                                                                                                                                                                                                                                                                                                                                                                                |
| IBM BigInsights 3   | Hive 2                                    | ./libs/ibm/hive/                                                             | commons-configuration-1.6.jar<br>commons-logging-1.1.1.jar<br>hadoop-core-2.2.0-mr1.jar<br>hive-exec-0.12.0.jar<br>hive-jdbc-0.12.0.jar<br>hive-metastore-0.12.0.jar<br>hive-service-0.12.0.jar<br>hive-shims-0.12.0.jar<br>httpclient-4.2.5.jar<br>httpcore-4.2.4.jar<br>libfb303-0.9.0.jar<br>libthrift-0.9.0.jar<br>log4j-1.2.17.jar<br>slf4j-api-1.6.1.jar<br>slf4j-log4j12-1.6.1.jar |
|                     | BigSQL                                    | ./libs/ibm/bigsql/                                                           | db2jcc.jar                                                                                                                                                                                                                                                                                                                                                                                |
|                     | BigSQLv1                                  | ./libs/ibm/bigsqlv1/                                                         | bigsql-jdbc-driver.jar                                                                                                                                                                                                                                                                                                                                                                    |

Table 3-1. List of JDBC drivers for the supported service providers (continued)

| Provider | Service          | NPS path                                        | Sample file list                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------|------------------|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cloudera | Impala/<br>Hive2 | ./libs/cloudera/hive/./<br>libs/cloudera/impala | <p>You can download drivers from the Cloudera website or download the following drivers from a Cloudera installation. See “Obtaining Cloudera JDBC drivers” on page 3-6 later in this topic.</p> <p> commons-collections.jar<br/> commons-configuration.jar<br/> commons-lang.jar<br/> commons-logging-1.0.4.jar<br/> guava.jar<br/> hadoop-annotations-2.0.0-cdh4.6.0.jar<br/> hadoop-auth-2.0.0-cdh4.6.0.jar<br/> hadoop-common-2.0.0-cdh4.6.0.jar<br/> hadoop-mapreduce-client-core.jar<br/> hive-exec-0.10.0-cdh4.6.0.jar<br/> hive-jdbc-0.10.0-cdh4.6.0.jar<br/> hive-metastore-0.10.0-cdh4.6.0.jar<br/> hive-service-0.10.0-cdh4.6.0.jar<br/> hive-shims-0.10.0-cdh4.6.0.jar<br/> httpclient-4.2.5.jar<br/> httpcore-4.2.5.jar<br/> libfb303-0.9.0.jar<br/> log4j-1.2.16.jar<br/> slf4j-api-1.6.4.jar<br/> slf4j-log4j12-1.6.1.jar </p> |

Table 3-1. List of JDBC drivers for the supported service providers (continued)

| Provider    | Service | NPS path            | Sample file list                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------|---------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Hortonworks | Hive2   | ./libs/horton/hive/ | <p>For Hortonworks Data Platform version 2.3:</p> <p>commons-codec-1.4.jar<br/> commons-collections-3.2.1.jar<br/> commons-configuration-1.6.jar<br/> commons-logging-1.1.3.jar<br/> hadoop-auth-2.4.0.2.1.7.0-784.jar<br/> hadoop-common-2.4.0.2.1.7.0-784.jar<br/> hadoop-mapreduce-client-core-2.4.0.2.1.7.0-784.jar<br/> hive-common-0.13.0.2.1.7.0-784.jar<br/> hive-exec-0.13.0.2.1.7.0-784.jar<br/> hive-jdbc-0.13.0.2.1.7.0-784.jar<br/> hive-metastore-1.2.1.2.3.0.0-2557.jar<br/> hive-service-0.13.0.2.1.7.0-784.jar<br/> httpclient-4.2.5.jar<br/> httpcore-4.2.5.jar<br/> libthrift-0.9.0.jar<br/> log4j-1.2.17.jar<br/> slf4j-api-1.7.5.jar</p> <p>For Hortonworks Data Platform versions prior to 2.3:</p> <p>commons-codec-1.4.jar<br/> commons-collections-3.2.1.jar<br/> commons-configuration-1.6.jar<br/> commons-logging-1.1.3.jar<br/> hadoop-auth-2.4.0.2.1.7.0-784.jar<br/> hadoop-common-2.4.0.2.1.7.0-784.jar<br/> hadoop-mapreduce-client-core-2.4.0.2.1.7.0-784.jar<br/> hive-common-0.13.0.2.1.7.0-784.jar<br/> hive-exec-0.13.0.2.1.7.0-784.jar<br/> hive-jdbc-0.13.0.2.1.7.0-784.jar<br/> hive-service-0.13.0.2.1.7.0-784.jar<br/> httpclient-4.2.5.jar<br/> httpcore-4.2.5.jar<br/> libthrift-0.9.0.jar<br/> log4j-1.2.17.jar<br/> slf4j-api-1.7.5.jar</p> |

The steps to obtain the JDBC files are different for each service provider. The following sections describe some general steps for obtaining the JDBC files for each vendor. See the “Troubleshooting missing JDBC drivers” on page 3-7 for general information about finding the JDBC files. Refer to the documentation for your Hadoop service provider software for specific details about locating and obtaining the JDBC driver client files.

## Obtaining IBM BigInsights 3 JDBC drivers

If you use the IBM BigInsights 3 BigSQL and Hive2 services, you can obtain the JDBC driver files as follows:

1. Log in to the BigInsights Web Console.
2. Click **Quick Links > Download client library and development software**.
3. Select the following software bundles:
  - Hive JDBC package

- BigSql and BigSqlv1 client libraries
4. Save the JDBC files to the directory shown in Table 3-1 on page 3-3.

## Obtaining IBM BigInsights 4 JDBC drivers

If you use the IBM BigInsights 4 BigSQL and Hive2 services, you can obtain the JDBC driver files as follows:

1. Copy the required JAR files (shown in Table 3-1 on page 3-3) from the Hadoop master node. The files are typically located in the following directories:
  - For Hive:
    - /usr/iop/4.0.0.0/hive/lib/
    - /usr/iop/4.0.0.0/hadoop/
    - /etc/hadoop/conf/
  - For BigSQL:
    - /usr/ibmpacks/bigsql/4.0/db2/java/
2. You can copy all of the JAR files if you want, but you need only the files shown in Table 3-1 on page 3-3.
3. Save the JDBC files to the directory shown in Table 3-1 on page 3-3.

## Obtaining Cloudera JDBC drivers

Drivers are available on the Cloudera website or you can download the driver files from a Cloudera installation package.

- If you choose to download drivers from the Cloudera web site:  
 Note that there are separate packages for Hive and Impala. The Hive package must be copied to `./libs/cloudera/hive/` and the Impala package to `./libs/cloudera/impala`. Each download packages has several JDBC driver zip files. Ensure that you use the JDBC4 (not JDBC41) zip file for the compatible drivers. For both, Impala and Hive, JDBC4 drivers are the only supported version. The other JDBC versions that are included in the package are not compatible with IBM Fluid Query.

**Note:** If you download and use the drivers from the Cloudera web site and encounter problems, you can use the drivers from a Cloudera installation kit as a workaround. Note that you must remove the drivers from the web download package first before you replace them with the other drivers.

- If you choose to obtain the JDBC driver files from a Cloudera installation, use the following steps:
  1. Copy the required JAR files (shown in Table 3-1 on page 3-3) from the Hadoop master node. The files are typically in the following directories:
    - /usr/lib/hive/lib/
    - /usr/lib/hadoop/client/
    - /opt/cloudera/parcels/CDH/lib/hadoop/lib
  2. You can copy all of the JAR files if you want, but you need only the files shown in Table 3-1 on page 3-3.
  3. Save the JDBC files to the directory shown in Table 3-1 on page 3-3.

### Important:

- For connections to the Impala service, the IBM Fluid Query features only support the Hive2 drivers.

- Cloudera 5.4 uses Java 7 JDBC drivers. You can only use Java 7 drivers with IBM Fluid Query 1.6 and IBM Netezza Analytics 3.2.1. If you are using IBM Netezza Analytics version prior to 3.2.1, download and use the Java 6 drivers from a Cloudera 5.3 installation kit.

## Obtaining Hortonworks JDBC drivers

If you use the Hortonworks Hive2 services, you could obtain the JDBC driver files as follows:

1. Copy the required JAR files (shown in Table 3-1 on page 3-3) from the Hadoop master node. The files are typically in the following two directories, where X.X.X.X-XXXX are the software version and build numbers for your service provider:

- ./usr/hdp/X.X.X.X-XXXX/hive/lib/
- ./usr/hdp/X.X.X.X-XXXX/hadoop/client/

For example:

- ./usr/hdp/2.2.0.0-2041/hive/lib/
- ./usr/hdp/2.2.0.0-2041/hadoop/client/

2. You can copy all of the JAR files if you want, but you need only the files shown in Table 3-1 on page 3-3.
3. Save the JDBC files to the directory shown in Table 3-1 on page 3-3.

**Important:** The JDBC drivers for Hortonworks version 2.3 are Java 7 drivers. You can only use Java 7 drivers with IBM Fluid Query 1.6 and IBM Netezza Analytics 3.2.1. If you are using IBM Netezza Analytics version prior to 3.2.1, download and use the Java 6 drivers from a Hortonworks 2.2 installation kit.

## Troubleshooting missing JDBC drivers

If you encounter any problems finding or obtaining the JDBC files listed in Table 3-1 on page 3-3, or if you see `ClassNotFoundException` error messages when trying to configure connections, you may be missing a JDBC file required for that service connection. Sometimes the files may be in other locations on the system. If you cannot find a JDBC file, you can try to search for the file on your service provider's master node.

1. Log in to the service provider's master node as the root user.
2. Run a command similar to the following, to search for the missing file:

```
find / -name "missing-jar-name*.jar"
```

For example:

```
find / -name "commons-configuration*.jar"
```

3. The command displays a pathname for the file if it is present on the system. If you cannot find the required JDBC file and the **fqConfigure.sh** script continues to show errors for missing files, contact your service provider for assistance with obtaining the missing file.

## Data type mapping for predefined connectors

The following tables present data type mapping between NPS and target systems or databases.

## NPS to NPS (IBM PureData System for Analytics to IBM PureData System for Analytics)

The NPS data types match one-to-one when transferring data between NPS systems.

**Note:** For the TimeTZ data type, milliseconds are not transferred.

## NPS to Hadoop or Apache Spark SQL

| Data type on NPS | Data type on Hadoop          |
|------------------|------------------------------|
| Boolean          | Boolean                      |
| Double           | Double, Real, Float, Decimal |
| Int8             | TinyInt                      |
| Int16            | SmallInt                     |
| Int32            | Int, Integer                 |
| Int64            | BigInt                       |
| Char(X)          | Char(X)                      |
| Varchar(X)       | String, CLOB, Varchar        |
| Timestamp        | Timestamp                    |
| Date             | Date                         |
| VarBinary(X)     | Binary                       |

**Note:** The complex data types such as Arrays, Map, STRUCT, and UnionType are not supported.

If you use Cloudera Impala services, note that the FLOAT data type is converted to Double due to Impala's automatic FLOAT to DOUBLE type conversion. The Impala DOUBLE is more precise than the NPS Double data type.

## NPS to IBM DB2, dashDB, or IBM PureData® System for Operational Analytics

| Data type on NPS | Data type on DB2, dashDB, or PureData System for Operational Analytics |
|------------------|------------------------------------------------------------------------|
| Char(x)          | Character(x)                                                           |
| Varchar(x)       | Varchar(x), Clob(x), Dbclob(x), Graphic(x), Vargraphic(x)              |
| Varbinary(x)     | Binary(x), Varbinary(x), Blob(x)                                       |
| Int16            | Smallint                                                               |
| Int32            | Int, Integer                                                           |
| Int64            | Bigint                                                                 |
| Numeric(p,s)     | Decimal / Numeric(p,s)                                                 |
| Float            | Real                                                                   |
| Double           | Double                                                                 |
| Date             | Date                                                                   |
| Time             | Time                                                                   |
| Timestamp        | Timestamp                                                              |



| Data type on NPS | Data type on DB2, dashDB, or PureData System for Operational Analytics |
|------------------|------------------------------------------------------------------------|
| Xml              | Varchar(x)                                                             |

## NPS to Oracle

| Data type on NPS | Oracle                                                |
|------------------|-------------------------------------------------------|
| Char(x)          | Char(x)                                               |
| Varchar(x)       | Varchar(x), Varchar2(x)                               |
| Nchar(x)         | Nchar(x)                                              |
| Nvarchar(x)      | Nvarchar2(x)                                          |
| Varchar(x)       | Clob, Number                                          |
| Numeric(p,s)     | Number(p,s)                                           |
| Date             | Date                                                  |
| Timestamp        | Timestamp, Timestamp with TZ, Timestamp with local TZ |
| Int32            | Int, Integer                                          |
| Double           | Float, Double, Double precision, Real                 |
| Varchar(x)       | Long                                                  |

IBM Fluid Query does not support the following data types:

- Complex types that map to oracle.jdbc.OracleStruct (User-created types, including HTTPURIType)
- Types that map to oracle.jdbc.OracleOpaque (including SYS.XMLTYPE)
- Type oracle.sql.BFILE is also not supported.

**Note:** The data type **NUMBER** (without precision and scale) that exists in Oracle is not available in NPS. That is why **NUMBER** without precision is mapped to **VARCHAR** on NPS. **NUMBER** with specified precision and/or scale, like **NUMBER(10)** or **NUMBER(10,2)**, is normally mapped to **NUMERIC** on NPS.

## Kerberos configuration file

If your service provider uses Kerberos authentication, you must have a Kerberos configuration file for establishing the connection from the NPS host.

The `krb5.conf` file specifies configuration parameters that are required for Kerberos authentication. In many Kerberos environments, the `krb5.conf` file is already available for the Kerberos client support. Consult with your Kerberos administrator to obtain a copy of the `krb5.conf` file that you can store on the NPS hosts.

The default location for the Kerberos configuration file is `/etc/krb5.conf`. If you store the file in a different location, make sure that you specify the path to the file using the `--krb5-conf` argument of the **fqConfigure.sh** script. Also make sure that you save the file to the same path on both NPS hosts (HA1 and HA2) so that the file is available if an NPS host failover occurs.

---

## Configuring a connection to any JDBC data source

To allow connection to any target database using a generic connector feature, you define all connection details in the XML file and then run the **fqConfigure** script.

### Before you begin

Refer to your data service provider documentation to find out which JDBC drivers must be installed, and the JDBC connection configuration details.

### Procedure

1. Download the JDBC drivers for the selected service provider and store them in any location on NPS. Take note of the file path.
2. Create a configuration XML file. You can use one of the templates available in the `connectorTemplates` folder under the Fluid Query installation directory on NPS, or create a new file.

- The following templates are available:

- Informix
- MySQL
- Oracle
- SQLServer
- TeraData
- PostgreSQL
- MapR-DB

In addition, the following data mapping templates are available:

- SQLServer
- TeraData

- If there is no template available for your connection, create the XML file with the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<connection>
  <driverClass></driverClass>
  <classPath></classPath>
  <connectionURL></connectionURL>
  <jdbc-properties>
    <property>
      <name></name>
      <value></value>
    </property>
  </jdbc-properties>
</connection>
```

The `<classPath>` parameter corresponds to the file path to JDBC drivers.

You can define a simple XML file with hardcoded values, but you can also use variables to create connection templates. Moreover, you can define data types mapping for your connector if required. The various configurations are described in detail in the following sections.

3. When the XML file is created, run the `fqConfigure.sh` script.

The following parameters are required for the script:

- `--provider generic`
- `--service generic`
- `--connectionXmlPath <filepath>`

The `--dataTypesXmlPath` is optional.

## Configuring a connection with hardcoded values

In this task, you create a `netezza.xml` configuration file with hardcoded values to connect to the Netezza database.

### Procedure

1. Create the XML file with the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<connection>
  <driverClass>org.netezza.Driver</driverClass>
  <classPath>/local/path/to/jdbc/drivers/</classPath>
  <connectionURL>jdbc:netezza://192.168.1.10:5480/sampleDB</connectionURL>
  <jdbc-properties>
    <property>
      <name>user</name>
      <value>netezzaUsername</value>
    </property>
    <property>
      <name>password</name>
      <value>somePasswordInPlainText</value>
    </property>
  </jdbc-properties>
</connection>
```

2. Save the file as `/nz/export/ae/products/fluidquery/netezza.xml`

3. Run the following command:

```
/nz/export/ae/products/fluidquery/fqConfigure.sh --provider generic
--service generic --connectionXmlPath /nz/export/ae/products/fluidquery/netezza.xml
--config sample_generic
```

## Configuring a connection with variables

In this task, you create a `netezza.xml` configuration file with variables.

### About this task

With the configuration XML file, you can define all standard JDBC properties. To allow even better flexibility and creating connection templates, IBM Fluid Query supports variables. Selected `fqConfigure.sh` parameters have a corresponding `VARIABLE_NAME`, that you can use to create a more generic XML file. Supported variables include:

Table 3-2. Variables supported in the connection configuration XML

FqConfigure parameter	Variable name
--username	USERNAME
--password	PASSWORD
--hostname	HOSTNAME
--port	PORT
--database	DATABASE
--driver-path	DRIVER_PATH
--ssl-truststore	SSL_TRUSTSTORE
--ssl-truststore-password	SSL_TRUSTSTORE_PASSWORD

### Procedure

1. Create the XML file with the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<connection>
  <driverClass>org.netezza.Driver</driverClass>
  <classPath>${DRIVER_PATH}</classPath>
  <connectionURL>jdbc:netezza://${HOSTNAME}:${PORT}/${DATABASE}</connectionURL>
  <jdbc-properties>
    <property>
      <name>user</name>
      <value>${USERNAME}</value>
    </property>
    <property>
      <name>password</name>
      <value>${PASSWORD}</value>
    </property>
  </jdbc-properties>
</connection>
```

2. Save the file as /nz/export/ae/products/fluidquery/netezza.xml

3. Run the following command:

```
[nz@fluidquerynps4 BVT]$ /nz/export/ae/products/fluidquery/fqConfigure.sh --provider generic
--service generic --connectionXMLpath /nz/export/ae/products/fluidquery/netezza.xml
--config test_conn --driver-path /nz/export/ae/products/fluidquery/libs/ibm/pda
--host 9.167.40.2 --port 5480 --database ifq --username admin --password password
```

**Note:** Values for all the variables that are present in the configuration file, except for PASSWORD, must also be provided in fqConfigure.

If no value is provided in the command line for variable PASSWORD, a password prompt is displayed.

Example 1:

```
<connectionURL>jdbc:hive2://${HOSTNAME}:${PORT}/default;ssl=true;sslTrustStore=/nzscratch/BVT/fluidquery/BVT_Horton_35_truststore;
trustStorePassword=changeit;</connectionURL>
<jdbc-properties>
  <property>
    <name>user</name>
    <value>${USERNAME}</value>
  </property>
  <property>
    <name>password</name>
    <value>${PASSWORD}</value>
  </property>
</jdbc-properties>
```

The required fqConfigure parameters are --host, --port, --username, and as an option --password. No other connection defining parameters can be provided.

Example 2:

```
<connectionURL>jdbc:hive2://${HOSTNAME}:1232/default;ssl=true;sslTrustStore=/nzscratch/BVT/fluidquery/BVT_Horton_35_truststore;
trustStorePassword=changeit;</connectionURL>
```

The only required fqConfigure parameter is --host, no other connection defining parameters can be used.

## Handling of the database parameter - \${DATABASE}

Learn how to handle the - \${DATABASE} parameter.

1. When the - \${DATABASE} is not present in the configuration XML file:

```
[nz@fluidquerynps1 fluidquery]$ cat /nzscratch/BVT/fluidquery/generic_xmls/BVT_Generic_Horton_hive_35.xml
<?xml version="1.0" encoding="UTF-8"?>
<connection>
  <driverClass>org.apache.hive.jdbc.HiveDriver</driverClass>
  <classPath>/nz/export/ae/products/fluidquery/libs/ Horton/hive/</classPath>
  <connectionURL>jdbc:hive2://${HOSTNAME}:10000/default;ssl=true;sslTrustStore=/nzscratch/BVT/fluidquery/BVT_Horton_35_truststore;
trustStorePassword=changeit;</connectionURL>
<jdbc-properties>
  <property>
    <name>user</name>
    <value>${USERNAME}</value>
```

```

    </property>
  </property>
  <name>password</name>
  <value>${PASSWORD}</value>
</property>
</jdbc-properties>
</connection>

```

In this case, the first parameter of **fqRead** must not be provided, otherwise an error is thrown:

```

KF_TEST.ADMIN(ADMIN)=> select * from table with final (fqread('ifq','', 'select * from ifq.teradata_types;'));
ERROR: Value for variable: ${DATABASE} provided but not used in configuration XML.

```

## 2. When the - **\${DATABASE}** parameter is present in the configuration XML file:

```

<connectionURL>jdbc:hive2://${HOSTNAME}:10000/${DATABASE};ssl=true;sslTrustStore=/nzscratch/BVT/fluidquery/BVT_Horton_35_truststore;
trustStorePassword=changeit;</connectionURL>

```

In this case, the **--database** parameter is mandatory. If it is not provided, **fqConfigure** fails with the following message:

```

[nz@fluidquerynps1 fluidquery]$ /nz/export/ae/products/fluidquery/./fqConfigure.sh --provider generic --service generic --connectionXmlPath
/nzscratch/BVT/fluidquery/generic_xmIs/BVT_Generic_Horton_hive_35.xml --host 9.167.40.35 --username hdfs --password hdfs --config kf_gen
Value not provided for variable: DATABASE
Connection configuration failure. Please check log file: /nz/export/ae/products/fluidquery/logs/ConnectionConfigurator_20151027T06:38:12.082.log
for details.

```

If configuration is successful with the **--database** parameter, then the provided database becomes the default database name for **fqread** invocations:

```

[nz@fluidquerynps1 fluidquery]$ /nz/export/ae/products/fluidquery/./fqConfigure.sh --provider generic --service generic --connectionXmlPath
/nzscratch/BVT/fluidquery/generic_xmIs/BVT_Generic_Horton_hive_35.xml --host 9.167.40.35 --username hdfs --password hdfs --config kf_gen
--database ifq
log4j:WARN No appenders could be found for logger (org.apache.hive.jdbc.Utils).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Connection configuration success.

```

You can use either the default or customized value for database in **fqread**. The following example uses default database, which is 'ifq':

```
select * from table with final (fqread('','', 'select * from customer'));
```

In the following example, a customized value for database is used:

```
select * from table with final (fqread('tpch','', 'select * from customer'));
```

## Configuring a connection with customized data types mapping

In this task, apart from creating a connection configuration file, you create another XML file to define data types mapping.

### About this task

IBM Fluid Query comes with a set of pre-defined default mapping (described in “Data type mapping for generic connector” on page 3-15), which you can extend by adding new mappings or overwrite the existing mappings. When using a generic connector feature, you can map target database data types to NPS specific types in the XML file.

### Procedure

1. Create the `/nz/export/ae/products/fluidquery/connectionsample.xml` file as in the previous tasks.

2. Create the /nz/export/ae/products/fluidquery/types.xml file, where you define data type mappings.

For each mapping, you must provide:

- sourceType – source column type name.
- targetType
  - name – Netezza data type name
  - precision – for types that require precision, like Varchar, Nvarchar, Var binary
  - scale – for types that require scale, like Numeric

Note that is not possible to map any data type to any other type, for example, to map Float to TimeZone. The type of Java object returned by JDBC driver for a given sourceType must be compatible with Netezza targetType. The following table shows all the supported Netezza data types, and compatible Java objects.

Table 3-3. Netezza data types and compatible Java objects.

Netezza Target type name	Compatible Java object
Varchar/NVarchar/VarBinary <b>Note:</b> In types.xml you must use the following: <ul style="list-style-type: none"><li>• Variable for Varchar</li><li>• National_variable for NVarchar</li></ul>	java.lang.String, byte[]
Character/NCharacter <b>Note:</b> In types.xml you must use the following: <ul style="list-style-type: none"><li>• Fixed for Character</li><li>• National_fixed for NCharacter</li></ul>	java.lang.String
Numeric128/Float/Double/Int8/Int16/Int32/Int64/Real	java.lang.Number
Date	java.sql.Date
Time	java.sql.Time
TimeStamp	java.sql.TimeStamp
Bool	java.lang.Boolean

**Note:** The following target Netezza data types are not supported: TimeTz, ST\_Geometry, Interval.

Sample types.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataMappings>
<mappings>
  <mapping>
    <sourceType>Long_string</sourceType>
    <targetType>
      <name>fixed</name>
    </targetType>
  </mapping>

  <mapping>
    <sourceType>Lstring</sourceType>
    <targetType>
      <name>fixed</name>
    </targetType>
  </mapping>
</mappings>
```

```

<mapping>
<sourceType>Nvarchar2</sourceType>
  <targetType>
    <name>NATIONAL_VARIABLE</name>
    <!-- If precision is missing, Fluid Query will use precision
    data from jdbc driver metadata -->
  </targetType>
</mapping>
<mapping>
  <sourceType>boolean</sourceType>
  <targetType>
    <name>bool</name>
  </targetType>
</mapping>
<mapping>
  <sourceType>Numeric</sourceType>
  <targetType>
    <name>NUMERIC128</name>
    <precision>18</precision>
    <scale>8</scale>
    <!-- Can provide custom precision&scale. If not, Fluid Query will use
    jdbc driver metadata -->
  </targetType>
</mapping>
</mappings>
</dataMappings>

```

3. Run the following command:

```

/nz/export/ae/products/fluidquery/fqConfigure.sh --provider generic --service generic
--connectionXmlPath /nz/export/ae/products/fluidquery/connections/sample.xml
--config sample_generic --username admin --password password
--dataTypesXmlPath /nz/export/ae/products/fluidquery/types.xml

```

## Data type mapping for generic connector

The table presents predefined data type mapping used for generic connector.

Table 3-4. Predefined data type mapping for generic connector

IBM PureData System for Analytics target types	JDBC type names
Boolean	Boolean, Sql_Boolean
Character	Char, Character, Sql_Char
Ncharacter	Nchar
Varchar	Varchar, Varchar2, Clob, Dbclob, Graphic, Vargraphic, String, Xml, Sql_Varchar. Note: All other unknown types
Nvarchar	Nvarchar, Nvarchar2, NvChar
Varbinary	Binary, Varbinary, Blob, Sql_Varbinary
Int8	TinyInt, Sql_TinyInt, ByteInt
Int16	Smallint, Sql_SmallInt
Int32	Int, Integer, Sql_Integer
Int64	Bigint, Sql_BigInt
Numeric	Number, Numeric
Double	Float, Double, Double_Precision, Sql_Double, Decimal, Sql_Decimal, Real, Sql_Real
Date	Date, Sql_Date
Time	Time

Table 3-4. Predefined data type mapping for generic connector (continued)

IBM PureData System for Analytics target types	JDBC type names
Timestamp	Timestamp, TimestampTz, Sql_Timestamp, Sql_Type_Timestamp

## Configuring a connection with Kerberos authentication

You can configure a connection using JDBC driver and use Kerberos for authentication.

### Before you begin

If your service provider uses Kerberos authentication, you must have a Kerberos configuration file called `krb5.conf` for establishing the connection from the NPS host. For more details see “Kerberos configuration file” on page 3-9. You must also have information on service principal and client principal.

To configure a connection with Kerberos, you follow the basic guidelines as described in “Configuring a connection to any JDBC data source” on page 3-10, and define Kerberos specific parameters. Note that you can also configure a connection with variables when using Kerberos.

### Procedure

1. Create a configuration XML file. Define paths and connection URL according to your environment setup. You must also define service principal if required. The following sample includes a Hive service principal.

```
<?xml version="1.0" encoding="UTF-8"?>
<connection>
  <driverClass>org.apache.hive.jdbc.HiveDriver</driverClass>
  <classPath>/nz/export/ae/products/fluidquery/libs/ibm/hive</classPath>
  <connectionURL>jdbc:hive2://192.168.0.1:10000/default;principal=hive/myhost.example.com@example.com;</connectionURL>
  <jdbc-properties>
  </jdbc-properties>
</connection>
```

2. Using command line, you run the `fqConfigure.sh` script where you must define client principal, a link to a Kerberos configuration file (optional) and either a keytab or password. The parameters are described in detail in Table 6-5 on page 6-5.

**Note:** Service principal that is listed in Table 6-5 on page 6-5 cannot be provided as a parameter for a generic connector - it must be written explicitly in the XML configuration file if it is required by the service (as in the sample).

```
/nz/export/ae/products/fluidquery/./fqConfigure.sh --provider generic --service generic
--connectionXmlPath /nz/export/ae/products/fluidquery/netezza.xml
--client-principal biadmin/oc1424046082.ibm.com@OC3681068626.IBM.COM
--krb5-conf /nzscratch/BVT/fluidquery/krb5_9.158.143.245
--keytab /nzscratch/BVT/fluidquery/biadmin.keytab --config genericKerberos
```

## Generic connector sample configurations

This section includes a few sample configurations that might help you understand how to configure connections.



## Sample Teradata configuration

Following is an example of the configuration XML file for Teradata.

```
<?xml version="1.0" encoding="UTF-8"?>
<connection>
  <driverClass>com.teradata.jdbc.TeraDriver</driverClass>
  <classPath>nzscratch/BVT/libs/generic/teradata</classPath>
  <connectionURL>jdbc:teradata://${HOSTNAME}/database=${DATABASE},CHARSET=UTF8</connectionURL>
  <jdbc-properties>
    <property>
      <name>user</name>
      <value>${USERNAME}</value>
    </property>
    <property>
      <name>password</name>
      <value>${PASSWORD}</value>
    </property>
  </jdbc-properties>
</connection>
```

In this example, the national characters support is configured in the following way:

1. JDBC session parameter is set in the JDBC connection string:  
CHARSET=UTF8
2. The mapping of text data is defined either by the --str-to-nvarchar argument to fqConfigure, or by custom mappings like the following:

```
    <mapping>
      <sourceType>VARCHAR</sourceType>
      <targetType>
        <name>NATIONAL_VARIABLE</name>
      </targetType>
    </mapping>
  <mapping>
    <sourceType>CHAR</sourceType>
    <targetType>
      <name>NATIONAL_FIXED</name>
    </targetType>
  </mapping>
```

Note that in order to use the database (1st) parameter of **fqread** in the following way:

```
select * from table with final (fqread('mydb','mytable'))
```

you must include the **\${DATABASE}** parameter in connection strings. The exact way of putting the database depends on a particular driver.

## Sample configuration for SSL

Following is an example of defining a connection string for Horton.

```
<?xml version="1.0" encoding="UTF-8"?>
<connection>
  <driverClass>org.apache.hive.jdbc.HiveDriver</driverClass>
  <classPath>nz/export/ae/products/fluidquery/libs/horton/hive</classPath>
  <connectionURL>jdbc:hive2://${HOSTNAME}:10000/default;ssl=true;sslTrustStore=nzscratch/BVT/fluidquery/BVT_Horton_35_truststore;
    trustStorePassword=changeit;</connectionURL>
  <jdbc-properties>
    <property>
      <name>user</name>
      <value>${USERNAME}</value>
    </property>
    <property>
      <name>password</name>
      <value>${PASSWORD}</value>
    </property>
  </jdbc-properties>
</connection>
```

## Sample configuration with Kerberos authentication

Following is a sample configuration XML file for Hive, it includes a Hive service principal (in bold):

```
<?xml version="1.0" encoding="UTF-8"?>
<connection>
  <driverClass>org.apache.hive.jdbc.HiveDriver</driverClass>
  <classPath>/nz/export/ae/products/fluidquery/libs/ibm/hive_BI3</classPath>
  <connectionURL>jdbc:hive2://9.167.40.33:10000/default;principal=hive/oc1424046082.ibm.com@OC3681068626.IBM.COM;</connectionURL>
  <jdbc-properties>
    </jdbc-properties>
</connection>
```

Sample configuration:

```
/nz/export/ae/products/fluidquery/./fqConfigure.sh --provider generic --service generic
--connectionXmlPath /nz/export/ae/products/fluidquery/netezza.xml
--client-principal biadmin/oc1424046082.ibm.com@OC3681068626.IBM.COM
--krb5-conf /nzscratch/BVT/fluidquery/krb5_9.158.143.245
--keytab /nzscratch/BVT/fluidquery/biadmin.keytab --config genericKerberos
```

## Sample MapR configuration

The configuration with MapR is described in detail in Chapter 7, “Appendix: Using IBM Fluid Query with MapR,” on page 7-1.

## Sample configuration for SAP HANA on Cloud

You can find detailed configuration steps required to connect to SAP HANA using a generic connector in a technote available at <http://www-01.ibm.com/support/docview.wss?uid=swg21986343>.

## Sample configuration for DB2/z

You can find detailed configuration steps required to access data in DB2/z in a technote available at <http://www-01.ibm.com/support/docview.wss?uid=swg21989853>.

---

## Encrypting password in SQL connector configuration file

To encrypt passwords in the connectors configuration files, you can either use the default static encryption, the auto-generated key, or create a key of your own.

### About this task

By default, the passwords provided when running the `fqConfigure.sh` script are encrypted in the configuration file with the use of a static key. To enhance security, starting with version 1.7.1, you can use the 128-bit AES key. The key can be automatically generated when running the `fqConfigure.sh` script, or you can generate your own key and store it in a file that you then provide when running the `fqConfigure` script.

### Procedure

- To use the auto-generated key for encryption, add the **--autoGenerateKey** parameter when running the `fqConfigure.sh` script:

```
./fqConfigure.sh --host <your_host> --provider horton --service hive --port 10000 --username root
--config <your_configuration_name> --autoGenerateKey
```

When running the `fqConfigure.sh` script you are prompted for password, which is then encrypted in the configuration file.

The encryption key file is stored in a default location `/nz/export/ae/products/fluidquery/AutoGenKeys/`, with a default name `autoGeneratedkey_<timestamp>`, and only the owner has the read permission to it.

You can also choose to store the file in a different location by adding the **--KeyFileOut** parameter:

```
./fqConfigure.sh --host <your_host> --provider horton --service hive --port 10000 --username root
--config <your_configuration_name> --autoGenerateKey --keyFileOut /tmp/keyfile
```

- To use the personal key for encryption:

1. Run the following command to generate a key. Specify the filepath to store the key file.

```
dd if=/dev/urandom of=<path/to/keyfile> bs=16 count=1
```

**Note:** When generating the key yourself, you are responsible for the security of the encryption key.

2. Use the **--keyfile** parameter when running the `fqConfigure.sh` script to specify the encryption key file that you generated:

```
./fqConfigure.sh --host <your_host> --provider horton --service hive --port 10000 --username root
--config <your_configuration_name> --keyFile /path/to/keyfile
```

---

## Registering the data connector functions

To call and use the data connector functions in your queries, you must register the data connector functions in the NPS database from which you plan to run the query.

### Before you begin

After you have created at least one connection to your external database service providers, you can then register the data connector functions in the NPS database. Typically, you register the functions only once in each NPS database that is used for SQL select queries with the external tables.

You can register the functions in local or remote mode. For more information see [About local and remote mode functions](#).

### Procedure

1. Log in to the NPS active host as the `nz` user.
2. Change to the `/nz/export/ae/products/fluidquery/` directory.
3. Run the **fqRegister.sh** script to register the functions. There are four data connector function definitions that have the same function name but different signatures to support different types of query invocations. By default, the data connector functions use the name `FqRead()`, but you can register the functions under a unique name for your environment. Sample commands follow.
  - To perform a simple registration where you add the `FqRead()` functions to the database specified by `NZ_DATABASE` (which is set to a database named `maindb`) using the connection defined in the `default.properties` configuration:

```
./fqRegister.sh
```

Functions and credentials are successfully registered in database "MAINDB".

- To register the functions using the name HdpRead() in the database named MyDb using the connection defined in the myConfig.properties connection configuration:

**Note:** When specifying a value for --config, do not include the .properties suffix of the filename.

```
./fqRegister.sh --udtf HdpRead --db MyDb --config myConfig
```

Functions and credentials are successfully registered in database "MYDB".

## Results

The registration function adds the data connector functions to the specified database. Each registration requires a connection configuration. You must specify a connection configuration file that was created by the **fqConfigure.sh** script, or the command uses the default connection called default.properties. The functions are owned by the admin user, and for other database users to use the functions in their queries, you must grant those users privileges to execute the functions. For more information about privileges, see “Assigning privileges to run the data connector functions” on page 3-27.

After registering the functions, your NPS users who have privileges to the database and to execute the functions can include them in their SQL queries. To confirm that the functions were added to the database, you can use the **SHOW FUNCTION** command to display the functions in your database. A sample command for a function registered with hdpreadname follows. Note that the ARGUMENTS column is truncated in the documentation because of the wide length of the field.

```
MYDB.ADMIN(MYUSER)=> SHOW FUNCTION hdpread;
```

SCHEMA	RESULT	FUNCTION	BUILTIN	ARGUMENTS
ADMIN	TABLE (ANY)	HDPREAD	f	(CHARACTER VARYING(ANY), CHARACTER VARYING(ANY))
ADMIN	TABLE (ANY)	HDPREAD	f	(CHARACTER VARYING(ANY), CHARACTER VARYING(ANY), CH...
ADMIN	TABLE (ANY)	HDPREAD	f	(CHARACTER VARYING(ANY), CHARACTER VARYING(ANY), CH...
ADMIN	TABLE (ANY)	HDPREAD	f	(CHARACTER VARYING(ANY), CHARACTER VARYING(ANY), CH...

(4 rows)

If you plan to use the remote mode for your data connector functions, you must register a data connector function with the --remote flag, otherwise the remote service will not start. See “Remote mode” on page 3-23 for more information.

If the command fails and displays an error message, you can try re-running command and adding the --debug option for more troubleshooting information. If you use the --debug option when you register the functions, the software creates log files each time a query runs and calls the functions. The log files can help you to troubleshoot any query problems, but when your queries are operating as expected, make sure that you re-register the functions without the --debug option to stop the log files. For more information about the script and its options, see “The fqRegister script” on page 6-8.

**Note:** The log file <FQ\_INST\_DIR>/logs/SqIRead\_<timestamp> is appended every time you run a query, regardless of the --debug option. It is important to clean up the /nz/export/ae/products/fluidquery/logs/ log directory periodically. If remote mode of SQL connector is used, remote mode daemon must be stopped before the cleanup task, and restarted after the cleanup task.

For more information about the FqRead() function and the four supported input forms, see “The FqRead function” on page 6-13.

## About local and remote mode functions

When you register the data connection functions, you can specify whether the functions run in local or remote mode.

Local mode is the default behavior for the data connector functions. To use remote mode, you must specify the `--remote` option when you register the functions using the **FqRegister.sh** script.

The data connector functions are built using the IBM Netezza Analytics feature called the Java Analytic Executables (also called user-defined analytic processes [UDAPs]). For Netezza Analytic Executables (AEs), the behavior mode specifies how the Java Virtual Machine (JVM) process handles the life cycle of each data connector function call when it runs. You can use the mode specification to select how you want the functions to run, which can help you to control the impacts on NPS resources.

### Local mode

By default, when you register a data connector function using the **fqRegister.sh** script, the data connector functions run using the local behavior mode.

For a local function, the NPS system controls the complete lifecycle of the function. When the query that calls a local function runs, the NPS system automatically launches the local function in a new process. In a successful run, the function processes input, produces output, and terminates normally. For unsuccessful runs, the NPS system terminates the function when the query finishes or has been terminated, and the function has not shut down.

In this model, the NPS system ensures that for each connector function call there is one running data connector function per dataslice. The NPS system does not allow an orphan function, that is, a function in which the query has ended, to keep running. If you run the Linux command **ps -H -e** while a data connector function is running, the function is shown as a child process of the NPS system process. Local AEs have a lifespan that is less than a query and technically less than a subset of a query. This life cycle model is similar to that of UDXs. A local AE always processes the input from exactly one SQL function call.

### Important considerations for local mode:

It is possible to grant query users direct access to data connector functions so that they can run any SQL query to retrieve rows from tables managed by the Hadoop service. However, you should not use local mode for this type of direct access.

If query users create queries to access the Hadoop tables directly, such as using a query similar to the following:

```
SELECT * FROM TABLE WITH FINAL ( fqread('mydb', '', 'select * from table_sample_07'
where id=123'));
```

This sample query causes two identical queries to run on the Hadoop service:

- The first query retrieves the metadata for the columns of the table and returns the metadata to the NPS side.
- The second query retrieves the data/records that match the query, processes the results as applicable, and returns the results to the NPS side.

Running these two queries could impact the performance of the Hadoop system, especially if the query is using more advanced analytics such as Map-Reduce

functions. To reduce the performance impact of these types of queries, it is recommended that the functions be registered as remote mode functions.

#### **Workload management considerations:**

Each query that calls the data connector functions, including each user query on the Hadoop-related NPS views, could cause a heavy resource utilization on the NPS system.

Each query results in a transfer of data from the Hadoop service to the NPS system, which is essentially a load operation in terms of resource usage and needs. As a best practice, NPS administrators should plan to limit the number of concurrent queries that use the Hadoop data connector functions on Hadoop-related views.

**Note:** As another alternative, consider using remote mode for your data connector functions to help reduce the impact on your NPS system.

#### **WLM settings for NPS releases before 7.1**

To limit resource usage on NPS systems that are running releases before 7.1, you can use the guaranteed resource allocation (GRA) controls to limit the number of concurrent jobs.

You could create a new resource group, and then set the Job Maximum attribute for the group to limit the number of concurrent jobs that run for the group. In addition, you could set the Resource Maximum attribute to limit the number of system resources that the group can use at any one time. Jobs that exceed the job maximum or the resource maximum limit will wait until the jobs or resources are available to run those queries. A sample SQL command follows:

```
CREATE GROUP analysts WITH RESOURCE MINIMUM 20 RESOURCE MAXIMUM 60 JOB MAXIMUM 3;
```

Assign all the users who have privileges to run the Hadoop data connector functions to the resource group so that the RA controls will manage the resources that they use and limit the effect on the NPS system. For example:

```
ALTER USER username IN RESOURCEGROUP analysts;
```

For more information on how to use and configure the workload management controls, see the GRA details in the *IBM Netezza System Administrator's Guide*.

#### **WLM settings for NPS releases 7.1 and later**

To limit resource usage on NPS systems that are running releases 7.1 and later, you can use the GRA resource groups as described in the previous section, and you can also use scheduler rules to limit the concurrent jobs and impacts on the system.

If your system uses user-defined functions (UDFs) only for the Hadoop data connector functions, you could create a scheduler rule to limit concurrent queries that call UDFs, for example:

```
CREATE SCHEDULER RULE FQRu1e AS IF TYPE IS UDX THEN LIMIT 1;
```

This rule allows only one query that calls a UDF to run at any time on the system. After that query finishes, the first queued query will then run, and so on.

For more information on how to use and configure scheduler rules, see the *IBM Netezza System Administrator's Guide*.

## Remote mode

When you register a data connector function using the **fqRegister.sh --remote** option, you configure the function to run using the remote behavior mode.

For a remote function, the NPS system does not control the life cycle of the remote function processes. If the NPS system is used to launch the remote mode function, it is launched daemon-style and disassociated from the NPS system process tree. A remote function may have a life cycle that ranges from less than a subset of a query up to indefinite, as with a long running daemon.

A remote function processes many SQL function calls simultaneously unless an instance exists per session or per dataslice and the function is invoked only once per SQL statement.

You can register Hadoop functions as remote only for JDBC-related services.

With remote mode, you can run queries for different types of services for each provider. Starting in Release v1.6, you can run queries for multiple services and versions at the same time.

## Recommended use

When running data connection functions in remote mode, you can use the functions in the following ways:

- For Hadoop views that will be used by query users
- For query users who plan to select and retrieve data directly from the Hadoop tables

In remote mode, the system runs only one query on the Hadoop service system, whereas two queries are run for local mode behavior.

**Important:** Read about limitations when using remote mode and Kerberos in “Troubleshooting problems with Kerberos” on page 5-16.

## Workload management considerations:

Remote mode functions also offer data connector configuration controls that you can use to help reduce impacts on the NPS system.

Remote mode functions include a configuration file that can limit the number of running data connector jobs in the system. The configuration file is stored at `/nz/export/ae/products/fluidquery/conf/fqConf.properties`. The file includes three settings that the NPS administrator can modify:

### **thread.active**

This setting controls the maximum number of concurrent data connector jobs that are actively running on the system. The default is 2. Users could launch any number of queries that call data connector functions, and the system launches the underlying data connector analytic executables up to the number of active threads. The system queues any additional data connector queries above the thread limit until a currently running job completes, then the system starts the next data connector executable. Data

connector queries could take a long time to complete if there are numerous other data connector queries waiting in the queue.

To help limit the number of queued data connector queries, you can use the `thread.queue` setting to limit the queue length. NPS does not have control over the queries executed by the data connector and it is important to limit the number of Hadoop queries that NPS can start. If you do not limit the Hadoop queries, you could encounter a situation where users think that their queries are running, and the NPS system has many queries listed as active, but only a few of the data connector queries are actually executing with the connector. The queued queries are waiting for the current queries to finish before they can start executing, so users would experience this as longer than expected query runtimes. In addition, too many concurrent queries could impact the overall workload of the NPS system and impact all NPS queries during that time. You can use resource groups and/or scheduler rules to help limit the number of NPS queries that can be started.

#### **thread.queue**

This setting controls the size of the data connector job queue. The default is 2. If more than the `thread.active` number of jobs are launched, any jobs above the number that can be started will be placed in a queue until they can be started. The queue size parameter specifies how many jobs can be queued while they wait to start. When the queue is full, any new jobs are automatically terminated.

#### **connection.timeout**

This setting controls the length of time in seconds to wait to establish a connection to the Hadoop service provider. The default is 10 seconds.

If you change the `fqConf.properties` configuration settings, you must stop and restart the data connector for the new settings to take effect.

### **Remote service administration tasks:**

When you use the data connector functions in remote mode, you must start and manage a remote service. The following sections discuss several administration tasks related to the operations of the remote service.

*Start the remote service:*

Before you run a data connector function that uses remote mode behavior, you must start the remote service.

You can start the remote service with the **fqRemote.sh** script or automatically when you run a remote mode function. The NPS software must be started before you can start the remote service.

To start the remote service using the script:

1. Log in to the Netezza active host as the `nz` user.
2. Run the following command:  
**./fqRemote.sh start**

To start the remote service using a function call:

1. Connect to an NPS database as a user who has privileges to run the following function.



2. Run the following query:

```
SELECT aeresult FROM TABLE WITH FINAL(inza..fq_launch(0));
```

Sample output follows for either the script or the function call.

```

                                AERESULT
-----
tran: 2704 session: 17646 DATA slc: 0 hardware: 0 machine: hostname
process: 24320 thread: 24321
```

If there are no remote mode functions registered on the NPS system when you start the remote service, the start operation returns the following error:

```
ERROR: Function 'FQ_LAUNCH(INT4)' does not exist
Unable to identify a function that satisfies the given argument types
You may need to add explicit typecasts
```

If the remote service is not running and a user runs a query that calls a remote mode Hadoop function, the query fails with an error message similar to the following:

```
select * from table with final (fqRead('','','select count(*) from test_table'));
ERROR: NzaeRemoteProtocolParent: timeout getting client connection (in file
nzaebasecontroller.cpp at line 1703)
```

*Stop the remote service:*

You can stop the remote service for troubleshooting or other administration tasks. When you stop the service, data connector functions will not run until they you start the remote service again.

You can stop the remote service manually with the **fqRemote.sh** script or automatically when you run a remote mode function.

To stop the remote service using the script:

1. Log in to the Netezza active host as the nz user.
2. Run the following command:

```
./fqRemote.sh stop
```

To stop the remote service using a function call:

1. Connect to an NPS database as a user who has privileges to run the following function.
2. Run the following query:

```
SELECT aeresult FROM TABLE WITH FINAL(inza..nzaejobcontrol('stopall', 0, NULL,
false, NULL, NULL));
```

Sample output follows for either the script or the function call.

```

                                AERESULT
-----
nzhost-H1 15576 (hadoopconnector dataslc:-1 sess:-1 trans:-1) AE stopped
(1 row)
```

*Test the remote service:*

You can test the remote service by using the **fqRemote.sh** script or automatically when you run a remote mode function to verify that the service is active and responding.

To test the remote service using the script:

1. Log in to the Netezza active host as the nz user.
2. Run the following command:  
`./fqRemote.sh ping`

To test the remote service using a function call:

1. Connect to an NPS database as a user who has privileges to run the following function.
2. Run the following query:  
`SELECT aeresult FROM TABLE WITH FINAL(inza..nzaejobcontrol('pingall', 0, NULL, false, NULL, NULL));`

The ping command displays information about the running processes, or if there are no processes running. For example, the following sample output shows that a remote service is running for the data connector, and displays the process ID (20831). The output information is abbreviated for the documentation example.

```

AERESULT
-----
nzhost-H1  20381 (hadoopconnector dataslc:-1 ... version: 10
(1 row)

```

The following sample output shows that the remote service is not running:

```

AERESULT
-----
(0 rows)(1 row)

```

Note that this test checks for the status of the remote service on the NPS host. It does not test or check the status of the remote Hadoop service provider.

*Display remote service process information:*

You can display process information about the remote service manually with the **fqRemote.sh** script or automatically when you run a remote mode function.

This operation can complete even if the remote service is not responding on the connection point. It returns data for all output columns that do not require direct communication with the remote service.

To display the remote service process information using the script:

1. Log in to the Netezza active host as the nz user.
2. Run the following command:  
`./fqRemote.sh ps`

To display the remote service processing information using a function call:

1. Connect to an NPS database as a user who has privileges to run the following function.
2. Run the following query:  
`SELECT aeresult FROM TABLE WITH FINAL(inza..nzaejobcontrol('psall', 0, NULL, false, NULL, NULL));`

Sample output follows for either the script or the function call.

```

AERESULT
-----
nzhost-H1  20477 (hadoopconnector ... /nz/export/ae/languages/java/java_sdk/ibm-java-i386-60/bin/java
(1 row)

```

*List the remote service connections:*

You can list all the data connections that are using the remote service with the **fqRemote.sh** script or automatically when you run a remote mode function.

To list the remote service connections using the script:

1. Log in to the Netezza active host as the nz user.
2. Run the following command:

```
./fqRemote.sh connections
```

To list the remote service connections using a function call:

1. Connect to an NPS database as a user who has privileges to run the following function.
2. Run the following query:

```
SELECT * FROM TABLE WITH FINAL(inza..nzaejobcontrol('connections', 0,  
null, false, null, null));
```

If the command output shows the message AE Stopped for any of the connections, that connection is hung. You can use the repair option to clean up the hung connections and release the resources. For more information about repairing the connections, see “Repair the remote service connection.”

*Repair the remote service connection:*

If you have a remote service connection that is hung and should be cleaned up, you can repair the connection on the NPS system.

The repair operation aborts any hung connections and cleans up leftover machine resources. This capability is useful in cases where a SQL data connector remote mode query fails or a session on the NPS appliance abruptly terminates.

To repair the remote service using the script:

1. Log in to the Netezza active host as the nz user.
2. Run the following command:

```
./fqRemote.sh repair
```

To repair the remote service using a function call:

1. Connect to an NPS database as a user who has privileges to run the following function.
2. Run the following query:

```
SELECT * FROM TABLE WITH FINAL(inza..nzaejobcontrol('repair', 0,  
null, false, null, null));
```

## **Assigning privileges to run the data connector functions**

To use the data connector functions in a SQL query, NPS database users must have privileges to run the functions.

### **About this task**

In each database where you register the data connector functions, the NPS administrator must enable the database with IBM Netezza Analytics privileges. You must also assign database users privileges to run the functions.

## Procedure

1. Log in to the NPS active host as the nz user.
2. Change to the /nz/export/ae/utilities/bin directory and run the following command to set up the Netezza Analytics operations, where *db* is a new or existing database on the system:

```
./create_inza_db.sh db
```

The command displays a series of messages to show that it creates a database (if it does not exist), schemas, and most importantly, three groups for the management of privileges related to the Netezza Analytics and data connector functions:

### ***db\_inzaadmins***

This is the group with local administration rights for this database, similar to what those users would have if they were the owner of the database.

### ***db\_inzadevelopers***

This group allows users to create and manage new AEs, UDXs, or stored procedures. Assign users to this group if they require privileges to create, alter, or drop the data connector functions in a database.

### ***db\_inzausers***

This group allows users to run or execute AEs, UDXs, or stored procedures. Assign users to this group if they require privileges to run queries that include the data connector functions.

3. To add a database user to the *db\_inzaadmins* group, run the following command:

```
./create_inza_db_admin.sh db username
```

This script must be run once per user per database. The specified user and database must exist.

4. To add a database user to the *db\_inzadevelopers* group, run the following command:

```
./create_inza_db_developer.sh db username
```

This script must be run once per user per database. The specified user and database must exist.

5. To add a database user to the *db\_inzausers* group, run the following command:

```
./create_inza_db_user.sh db username
```

This script must be run once per user per database. The specified user and database must exist.

## Results

When you complete this task, database users should be able to run SQL queries that include the data connector functions. If users create new data connector functions, either in new or different databases, you must repeat these steps to enable the database and other users to run those queries.

## Revoking privileges to run the data connector functions

For troubleshooting purposes, you can disable the IBM Netezza Analytics functions in a database, and you can also revoke privileges from users so that they cannot run SQL queries that include the data connector functions.

## About this task

In each database where you registered the data connector functions and configured IBM Netezza Analytics support, you can disable the Netezza Analytics support in that database for troubleshooting reasons or if the privileges were accidentally applied to the wrong database. When you disable the support, the process drops the three user groups that were created when you set up the support in “Assigning privileges to run the data connector functions” on page 3-27.

You can also selectively revoke privileges for a specific user.

## Procedure

1. Log in to the NPS active host as the nz user.
2. If you want to completely disable Netezza Analytics support in a database, change to the `/nz/export/ae/utilities/bin` directory and run the following command where *db* is the existing database on the system:

```
$/revoke_inza_db.sh db
```

The command displays a series of messages to show that it dropping the Netezza Analytics groups *db\_inzaadmins*, *db\_inzadevelopers*, and *db\_inzausers*, thus revoking privileges to manage, develop, and run the data connector functions.

3. If you are not disabling the Netezza Analytics support for a specific database, but you want to remove privileges for a specific user, you can do one of the following actions:
- To remove a database user from the *db\_inzaadmins* group, run the following command:

```
$/revoke_inza_db_admin.sh db username
```

- To remove a database user from the *db\_inzadevelopers* group, run the following command:

```
$/revoke_inza_db_developer.sh db username
```

- To remove a database user from the *db\_inzausers* group, run the following command:

```
$/revoke_inza_db_user.sh db username
```

- To revoke Execute privileges for the data connector functions from a user or group, connect to the database where the functions were registered as the database admin user, database owner, or as a member of the *db\_inzaadmins* group and run the following commands, specifying the function name that you registered:

```
revoke execute on function_name (varchar(any),varchar(any))  
    from user_name;  
revoke execute on function_name (varchar(any),varchar(any),  
    varchar(any)) from user_name;  
revoke execute on function_name (varchar(any),varchar(any),  
    varchar(any),varchar(any)) from user_name;  
revoke execute on function_name (varchar(any),varchar(any),  
    varchar(any),integer) from user_name;
```

**Tip:** If you had assigned privileges by assigning users to a group, such as a custom group or the *db\_inzausers* group, you can also just remove the user from that group to revoke privileges.

## Results

When you complete this task, the Netezza Analytics features should be disabled in the database if you ran the **revoke\_inza\_db.sh** script. Otherwise, if you ran the revoke commands for specific users, those users should no longer have those privileges. After the troubleshooting tasks are over, you can restore the Netezza Analytics support to a database and restore privileges as needed using the instructions in “Assigning privileges to run the data connector functions” on page 3-27.

---

## Using data connectors

Use the data connector user-defined table functions to create SQL queries that can select from structured data stored on the external file systems.

### Running SQL queries using the data connector

After you register your data connector functions in one or more databases, you can use the functions in your SQL queries to read data from tables that are stored on the Hadoop provider.

#### Before you begin

To run SQL commands that use the data connector functions, your database user account must have access privileges to the database where the functions are registered and privileges to execute the data connector functions in that database. See “Assigning privileges to run the data connector functions” on page 3-27. You must also have information about the Hadoop table that you want to query. The Hadoop database, table, and column names might be case-sensitive with the JDBC drivers, so make sure that you have the correct letter casing.

#### Procedure

1. Connect to an NPS database as a database user who has privileges to execute the data connector functions.
2. Create a SQL SELECT query to call the data connector function that you registered in the database. Some sample commands follow:

```
MYDB.ADMIN(MYUSR)=> SELECT * FROM TABLE WITH FINAL ( hdpread('', 'bidb.tab1'));
```

COL1	COL2
1	red
2	yellow
3	green
4	blue

```
MYDB.ADMIN(MYUSR)=> SELECT * FROM TABLE WITH FINAL ( hdpread('', '', 'select  
sum(col1) from bidb.tab1'));
```

1
10

#### Example

When you query tables that are managed by the Hortonworks Hive service, note that the Hive default behavior is to show column names in the table.column name format. This is a Hive setting that helps to establish unique column names. For example, a sample query that reads from a Hortonworks Hive table follows:

```
MYDB.MYSCH(MYUSR)=> SELECT * FROM TABLE WITH FINAL (fcread('mydb', 'employees'));
EMPLOYEES.EMP_ID | EMPLOYEES.NAME | EMPLOYEES.SALARY | EMPLOYEES.ADDRESS
-----+-----+-----+-----
          1 | Luis          |          10 | xyz
          2 | Mike          |          12 | abc
          3 | Joe           |          11 | xis
...

```

As shown in the sample output, the table name employees appears in each column name. You can configure the Hive service to omit the table name by setting the `hive.resultset.use.unique.column.names` property to false. For more information about this setting and how to edit it, refer to your Hortonworks documentation.

To select a particular column, remember to enclose the column name in quotations:

```
MYDB.MYSCH(MYUSR)=>select "EMPLOYEES.EMP_ID", "EMPLOYEES.NAME" from
table with final (fcread('mydb','employees'));
EMPLOYEES.EMP_ID | EMPLOYEES.NAME
-----+-----
          1 | Luis

```

If you omit the quotations around the column name, NPS uses the column name as a table name and returns an error:

```
MYDB.MYSCH(MYUSR)=> select EMPLOYEES.EMP_ID, EMPLOYEES.NAME from
table with final (fcread('mydb','employees'));
ERROR: relation does not exist MYDB.MYSHC.EMPLOYEES

```

Note that column names are case-sensitive, therefore the following query fails:

```
MYDB.MYSCH(MYUSR)=> select "EMPLOYEES.EMP_ID", "EMPLOYEES.name" from table with
final (fcread('mydb','employees'));
ERROR: Attribute 'EMPLOYEES.name' not found

```

If your query selects columns that have the same name, the query could fail with a column name error:

```
MYDB.MYSCH(MYUSR)=> SELECT * FROM TABLE WITH FINAL (fcread('', '', 'select parts.c1,
orders.c1 from parts join orders on parts.c1 = orders.c1'));
ERROR: Column already exists with same name

```

To avoid the column name error, specify a unique alias for any duplicate column names, as in the following example:

```
MYDB.MYSCH(MYUSR)=> SELECT * FROM TABLE WITH FINAL (fcread('', '', 'select parts.c1,
orders.c1 as c2 from parts join orders on parts.c1 = orders.c1'));
C1 | C2
----+----
23 | 23

```

If your query selects on values that require single quotation characters, make sure that you escape the quotation character by typing it twice as shown for the date value in the following example:

```
select * from TABLE WITH final (fcread('', '', 'select * from tbl where date_col =
''2015-07-02'' '));

```

For more information about the `FqRead()` function and the four supported input forms, see “The `FqRead` function” on page 6-13.

## Using views to simplify user queries

To simplify the SQL queries for popular Hadoop tables in your environment, NPS administrators can create views that call the data connector functions.

Users can create SQL queries as described in “Running SQL queries using the data connector” on page 3-30 to query tables in their Hadoop service providers, but those queries require information about the Hadoop tables and the data connector functions. The following steps show how an NPS administrator could create a view for use by query users and assign privileges to that view.

1. Log in to the NPS database and schema, if applicable, where the data connector functions are registered as a user who has privileges to create views.
2. Run the following command to create a view that calls the data connector query that connects to the Hadoop service and `table_sample_07` table to retrieve metadata about columns and their data types:

```
CREATE OR REPLACE VIEW SampleHadoopView AS SELECT * FROM TABLE WITH  
FINAL ( fqread('mydb', 'table_sample_07'));
```

3. Grant users or groups of users, as applicable, privileges to the view. The administrative user can run this command as needed for the applicable users or groups of user. For example:

```
GRANT SELECT ON SampleHadoopView TO user1;  
GRANT SELECT ON SampleHadoopView TO hdpgroup;
```

After the view is created and the users are granted privileges to select from the view, query users such as `user1` or the users in `hdpgroup` can use the `SampleHadoopView` to query the Hadoop table without needing to know all the specific information for the data connector functions, for example:

```
SELECT * FROM SampleHadoopView;
```

---

## Unregistering the data connector functions

If you no longer use a data connector function, you can unregister it to remove it from your NPS database.

### Before you begin

The unregister process removes the functions from a specified database. Before you begin, make sure that you have the database name and the name of the function that you want to remove.

### Procedure

1. Log in to the NPS active host as the `nz` user.
2. Change to the `/nz/export/ae/products/fluidquery/` directory.
3. Run the **`fqRegister.sh`** script with the `--unregister` and `--udtf` options to define which functions to unregister. For example, to unregister the function named `HdpRead()` in the database named `MyDb`, run the following command:

```
./fqRegister.sh --unregister --udtf HdpRead --db MyDb  
Functions and credentials unregistered successfully from database "MYDB".
```

If you do not specify the `--udtf` parameter, the last registered function for a given config is unregistered. When no config is specified, the default is used.

### Results

After unregistering the functions, any NPS queries or scripts that use the specified functions no longer work.



If the command cannot find the functions in the specified database, the command displays errors similar to the following output. Check the function name and database to make sure that you are supplying the correct information.

```
ERROR: ResolveRoutineObj: function 'HDPREAD(VARCHAR, VARCHAR)' does not exist
ERROR: ResolveRoutineObj: function 'HDPREAD(VARCHAR, VARCHAR, VARCHAR)' does not exist
ERROR: ResolveRoutineObj: function 'HDPREAD(VARCHAR, VARCHAR, VARCHAR, VARCHAR)' does not exist
ERROR: ResolveRoutineObj: function 'HDPREAD(VARCHAR, VARCHAR, VARCHAR, INT4)' does not exist
Unregister operation for database "MYDB" failed.
```

If the command fails and displays an error message, you can try re-running command and adding the `--debug` option for more troubleshooting information. For more information about the script and its options, see “The `fqRegister` script” on page 6-8.



---

## Chapter 4. Data movement

IBM Fluid Query enables you to quickly transfer your data between your Hadoop and NPS environments.

There are two ways in which data can flow between NPS and Hadoop:

- **Import**

Data transfer from NPS to Hadoop.

Prior to IBM Fluid Query version 1.7, data import could occur in three modes:

- Text mode - The NPS tables are transferred in text format and saved to HDFS in text format.
- Mixed mode - The NPS tables are transferred in compressed format and saved to HDFS in text format.
- Compressed mode - The NPS tables are transferred in compressed format and saved to HDFS in compressed format.

**Restriction:** You can both store and query compressed data on Hadoop, however querying them requires further configuration. Refer to “Querying compressed NZBAK files from Hadoop” on page 4-43.

These modes are configured in the **Data format properties** section of the XML configuration files.

With version 1.7, you can import data files from NPS and store them on Hadoop in one of the following formats: Parquet, Avro, ORC, RCFile. To handle this type of conversion, a new property was introduced in the XML configuration file:

**nz.fq.data.format**. You can also use this property in combination with compression properties to define text, mixed and compressed modes of import that were used in previous releases. For more information see “Data format properties combinations” on page 4-21.

- **Export**

Data transfer from Hadoop to NPS.

You can export to NPS:

- Data files that were previously imported from NPS.
- Hadoop data files in text format.
- Hadoop data types in Hadoop-specific formats ( Parquet, Avro, ORC, RCFile) that were previously imported from NPS.
- Hive tables that were not previously imported from NPS.

You can run the import and export operations from NPS, from Hadoop, or from any other systems that run Java. Depending which system you plan to run the operations from, different configuration and execution steps apply. These different scenarios are described in the following sections.

### Notes on data movement feature

1. The data movement feature transfers data directly between Hadoop data nodes and NPS. Therefore, a working network connection between all data nodes and NPS is a prerequisite.
2. The data movement feature is supported on systems delivered by BigInsights, Hortonworks, and Cloudera.

3. Running the import and export operations from NPS or any other systems is only possible when HDFS is used. If BigInsights uses GPFS, you can only initiate import and export operations from Hadoop.
4. Data types mappings are static. They are described in “Data type mapping for data movement” on page 4-46.
5. Data movement from and to NPS with Kerberos authentication can only be started from Hadoop. Execution from NPS is not supported.
6. Data movement with the same destination table executed in parallel is not supported.

---

## Data movement algorithm

Depending on the type of data that you transfer and the direction of transfer (import or export), two different processes are used for data movement.

- Data that is in text or NZBAK format and is not partitioned or clustered can be moved in both directions using *a simple algorithm*, where data is moved to the target directory directly. Note that this is the only algorithm that is supported by BigSQL for data movement.
- Data that is stored in a Hadoop-specific format, or in Hive partitioned or clustered tables, is transferred in both directions using *a two-stage algorithm* that utilizes a Hive engine. An INSERT SELECT operation executed on Hive is a part of this process.

### Data import

- The simple algorithm, where the data files are transferred directly to HDFS and stored there in their original format, is used for data in text and NZBAK format, and tables that are not partitioned or clustered.
- The two-stage algorithm is used for Hadoop specific formats, Hive partitioned and clustered tables:
  1. First, data is sent in text format to HDFS, where a temporary table is created.
  2. Then, data is converted from text to Hadoop format or a partitioned/clustered table and the text data is deleted.

Note that you can use the **nz.fq.debug.keep.tmptable = true** parameter to keep the temporary text table for further analysis, for example, when troubleshooting. For more information see “Verifying intermediate text data files” on page 5-14

There is also a possibility to write data directly to the selected Hadoop format or partitioned tables by setting a property **fq.data.directmovement = true**, but such setting might entail a performance drop. For more information see “Enforcing direct data movement” on page 1-10.

### Data export

- The simple process is used for text and NZBAK formats - only for data previously imported to Hadoop using the simple algorithm, or for text files stored in HDFS directory provided in `fq.input.path` parameter. The files are loaded directly to NPS.
- The two-stage process is used for data imported previously using the two-stage process, and for Hive tables provided in `fq.hive.tablename` parameter, unless they were imported from NPS using the simple algorithm and their metadata is present in HDFS directory. The two-stage process includes the following steps:
  1. Data is converted to text format using Hive engine. A temporary Hive table is created and inserted from the source Hive table.
  2. Text data is loaded to NPS.

There is also a possibility to use direct transfer - in such case a temporary table is based on custom format, implemented as data loader to NPS. For more information see “Enforcing direct data movement” on page 1-10.

---

## Support for Hadoop-specific file formats

Starting with IBM Fluid Query version 1.7 you can import the data files from NPS and store them in one of the following Hadoop-specific file formats: Parquet, Avro, ORC, RCFile. Moreover, any tables that were previously imported from NPS can also be exported.

**Note:** There are several limitations to the functionality. Many of the limitations a consequence of Hive and Parquet/Avro/ORC/RCFile dependency. It is strongly recommended to verify data consistency after import to Hadoop formats. Unlike for text, mixed and nzbak transfer, data consistency cannot be guaranteed for Hadoop formats because of these dependencies. For a list of known limitations, read “Troubleshooting import and export of Hadoop formats” on page 5-12.

You can run data movement both from Hadoop and from NPS, but in both cases you must use a JDBC driver to connect to a Hive server. You can use the remote import (or export) configuration XML templates to configure data movement with new formats.

You specify one of Hadoop formats using the **nz.fq.data.format** property.

You can also select the type of compression for the format of your choice using **nz.fq.output.compressed** property.

Mixed mode of transfer is not supported for the Hadoop-specific formats, so when using these formats, you must either delete the **nz.fq.compress** property, leave it empty or set it to false.

Detailed configuration steps for Hadoop formats are described in “Preparing the configuration XML file for the Hadoop file formats” on page 4-20.

When you do not want to use the new formats, you can leave the **nz.fq.data.format** property empty. You can also use the **nz.fq.data.format** parameter with Netezza specific formats (those used prior to version 1.7), but then you must set the compression properties appropriately. All possible combinations are described in detail in “Data format properties combinations” on page 4-21.

---

## Checksum functionality for data movement

The checksum functionality allows you to check data consistency after moving data with IBM Fluid Query. This functionality was introduced in version 1.7.1.

IBM Fluid Query provides the capability of bulk data movement. You can transfer data between NPS and Hadoop with high speed.

Data integrity is one of the most important aspect of data migration. Data must be protected against unintended modification. With IBM Fluid Query checksum functionality, users can ensure that migrated data is consistent and no unintended changes were introduced during migration.

After the migration process, the checksum is calculated on both, NPS and Hadoop, with a single SQL query against the migrated table on each. If two data sets have

different checksums, then these data sets are different. If the checksums are the same, then there is high probability that the data sets contains exactly the same data.

In general, the checksum is calculated based on the following formula:

$$Cks = \left( rctn + \sum_{r=1}^{rcnt} \left( \sum_{c=1}^{ccnt} tobigint(v(c,r)) \right) \% \pi \right) \&(1^{64} - 1)$$

where:

- ccnt – number of columns
- rcnt – number of rows
- tobigint(...) – function which converts data from their type to bigint (see next chapter).
- % - modulo operation.

Example on NPS:

```
SELECT
  SUM(MOD(CAST(( 0::BIGINT +
    NVL2 ( "C1", "C1", -1 ) +
    NVL2 ( "C2", LENGTH ( "C2" ), -2 ) +
    NVL2 ( "C3", "C3" - '2000-01-01', -3 ) +
    NVL2 ( "C6", CASE WHEN "C6" IS TRUE THEN 1 ELSE 2 END, -5 )
    ) as bigint ), 3.1415926535898::numeric(14,13))) as checksum
FROM "T10A"
```

You can configure the checksum functionality in the import or export XML configuration files, by setting the `fq.checksum` and `fq.checksum.columns` properties. This is described in detail in “Using checksum to check data consistency” on page 4-25.

## Limitations

- Checksum calculation fails if the column names are different in source and target table.
- Checksum cannot be calculated properly if you use data type keywords as column names, for example INTEGER, DECIMAL etc.

---

## Preparing configuration XML files

To configure data movement, you can use the XML template files which contain all the required parameters and some default settings.

### About this task

If you plan to run the data movement feature from the Hadoop service provider, all of the configuration for the data movement feature is performed on the Hadoop side. If you plan to allow users to run the data movement feature from NPS or other systems, you must configure those systems to obtain the required files and configure the connections to the Hadoop service provider.

Prior to version 1.7.1, IBM Fluid Query supported two modes of data movement: local and remote mode. Local mode regards data movement, both import and export, initiated from the Hadoop system on which the moved data is stored.

Remote mode regards data movement from any other machine, such as NPS, another Hadoop instance, or any other system running Java. To use remote mode, additional parameters must be provided in the configuration XML files. With these parameters, IBM Fluid Query can use a JDBC driver to connect to Hive or BigSQL. Since the local mode has many limitations, in version 1.7.1 it is deprecated, and you only get the remote XML configuration templates with clean installation. However, you can still use the local mode if necessary after upgrading. The local mode templates are not removed after upgrading to 1.7.1.

The following XML file templates are available in version 1.7.1:

- fq-import-remote-conf.xml
- fq-export-remote-conf.xml

The following additional templates were available prior to version 1.7.1:

- fq-import-conf.xml
- fq-export-conf.xml

These two templates do not contain the section regarding SQL connection settings.

## Procedure

1. Copy the template file and save it with a different name. Remember that each configuration file defines a single transfer job, that is single or multiple tables from or to a particular NPS database.
2. Review and edit the required parameters, as described in “Data import configuration XML” or “Data export configuration XML” on page 4-14

**Tip:** You can always overwrite the parameters that are set in the configuration XML file using a command line when running import or export. These options are described in “Running data movement from Hadoop” on page 4-28 and “Running data movement from NPS systems” on page 4-32.

3. Save the configuration file.

## Data import configuration XML

Learn how to customize the XML file which configures importing data from NPS to Hadoop.

### General configuration

- The **nz.fq.command** property sets the type of data movement: import (NPS->Hadoop) or export (Hadoop->NPS).

```
<property>
  <name>nz.fq.command</name>
  <value>import</value>
</property>
```

### HDFS information

- The **nz.fq.output.path** property sets the directory on HDFS where the transferred data is stored.

```
<property>
  <name>nz.fq.output.path</name>
  <value>/nzbackup/backup1</value>
</property>
```

- The **nz.fq.format.fielddelim** property sets the ASCII value of the single character field delimiter in the plain text output file.

```
<property>
  <name>nz.fq.format.fielddelim</name>
  <value>124</value>
</property>
```

**Note:** It is not recommended to set value **32** (space) as the value in this property.

- The **nz.fq.format.null** property defines how the NULL value will be represented. Default is NULL. The value cannot be longer than 4 characters.

```
<property>
  <name>nz.fq.format.null</name>
  <value>NULL</value>
</property>
```

- The **nz.fq.fs.temp** property sets the location of temporary files (such as logs and status files) on HDFS.

```
<property>
  <name>nz.fq.fs.temp</name>
  <value>/tmp</value>
</property>
```

- The **nz.fq.hive.schema** property sets the target schema name in Hive under which all imported tables are created.

```
<property>
  <name>nz.fq.hive.schema</name>
  <value></value>
</property>
```

If the property does not exist, it is automatically created. If the property is not set, the default schema is used.

- The **nz.fq.hive.tablename** property is optional. It defines the output table name in Hive. If not provided, table name from NPS is used. Use this property especially when importing a table with special characters in its name to Hive version lower than 0.13.0.

```
<property>
  <name>nz.fq.hive.tablename</name>
  <value>Test</value>
</property>
```

## Compression options

**Note:** A detailed description of all possible combinations of the format and compression properties, and the resulting conversion is described in “Data format properties combinations” on page 4-21

- The **nz.fq.compress** property defines whether to transfer NPS data in compressed internal format. For Hadoop-specific formats (PARQUET, ORC, RCFILE, and AVRO) data transfer is always processed in text, therefore this parameter must not be set to true.



```
<property>
  <name>nz.fq.compress</name>
  <value>true</value>
</property>
```

- The **nz.fq.output.compressed** property defines whether the transferred data is stored on Hadoop in compressed format. It can be set to true or false when **nz.fq.data.format** is deleted or empty. If this property is used with **nz.fq.data.format** with Hadoop-specific formats, depending on the format that you use, select one of the following values:
  - PARQUET: Snappy, gzip, uncompressed
  - ORC: NONE, ZLIB, SNAPPY
  - RCFILE: The value has to contain the exact class name of the codec which is available on Hadoop system. For example:  
org.apache.hadoop.io.compress.SnappyCodec
  - AVRO: snappy, deflate

You can also set the property to false or leave it empty, and the default compression type that is specified on Hadoop for the selected format will be used.

```
<property>
  <name>nz.fq.output.compressed</name>
  <value>>false</value>
</property>
```

- The **nz.fq.data.format** defines in which format the transferred data is stored on Hadoop. Supported values for this property are:
  - TXT, NZBAK - for Netezza-specific formats.
  - PARQUET, ORC, RCFILE, AVRO - for Hadoop-specific formats.

```
<property>
  <name>nz.fq.data.format</name>
  <value>TXT</value>
</property>
```

## Checksum options

- The **fq.checksum**

```
<property>
  <name>fq.checksum</name>
  <value>NONE</value>
</property>
```

- The **fq.checksum.columns**

```
<property>
  <name>fq.checksum.columns</name>
  <value></value>
</property>
```

## NPS information

- The **nz.fq.nps.db** property sets the NPS database name. Include double quotations around delimited database names.

```
<property>
  <name>nz.fq.nps.db</name>
  <value>dev</value>
</property>
```

- The **nz.fq.tables** property provides a comma-separated list of NPS tables that are to be imported. Include double quotations around delimited table names.

```
<property>
  <name>nz.fq.tables</name>
  <value>ADMIN.tab</value>
</property>
```

The format of this value is **<SCHEMA>.<table>**. You can use an asterisk (\*) character in the **nz.fq.tables** property as a filter, for a combination of multiple schemas and tables, for example:

- S1.T1 - imports table T1 in schema S1
- S1.\* - imports all tables from schema S1
- \*.T1 - imports all tables T1 from all schemas
- \*.\* - imports all schemas with all tables
- \* - imports all schemas with all tables

Note that you cannot use \* as part of the schema or table name and be used as a filter, for example: S1.a\*, meaning "all tables in schema S1, with name starting with a". The system cannot interpret such a filter, and a\* is interpreted as the exact table name.

**Restriction:** If full schema support is enabled on your NPS system, make sure to provide the schema name together with the table name in this property.

- The **nz.fq.nps.server** property sets the wall IP address or the fully qualified host name of the NPS server.

```
<property>
  <name>nz.fq.nps.server</name>
  <value>hostname.ibm.com</value>
</property>
```

- The **nz.fq.nps.port** property sets the port number for the NPS database instance NZ\_DBMS\_PORT.

```
<property>
  <name>nz.fq.nps.port</name>
  <value>5480</value>
</property>
```

- The **nz.fq.nps.user** property sets the NPS database user account name for access to the database.

```
<property>
  <name>nz.fq.nps.user</name>
  <value>admin</value>
</property>
```

- The **nz.fq.nps.password** property sets the password for the NPS database user account.

```
<property>
  <name>nz.fq.nps.password</name>
  <value>password</value>
</property>
```

- The **nz.fq.nps.ssl** property sets the NPS server connection type. When set to true, then only Secured JDBC mode is used. Default is false.

```
<property>
  <name>nz.fq.nps.ssl</name>
  <value>>false</value>
</property>
```

- The **nz.fq.nps.ssl.cacertificate** property sets the full path to the CA Certificate file that is stored on HDFS and used to authenticate connections. Used only when the SSL flag is true. If not provided, then all connections are accepted.

```
<property>
  <name>nz.fq.nps.ssl.cacertificate</name>
  <value></value>
</property>
```

- The **nz.fq.nps.where** property specifies the SQL WHERE clause that is used for selecting the data to transfer. This property only works for import, not export.

```
<property>
  <name>nz.fq.nps.where</name>
  <value></value>
</property>
```

**Restriction:** The **nz.fq.nps.where** parameter cannot contain a subquery if an import operation is performed in binary or mixed mode (**nz.fq.compress=true**). For instance, **nz.fq.nps.where=code in (select c\_custkey from utm\_hive\_116..customer)** will fail in mixed or binary mode with error similar to: Unable to read data from NPS: ERROR: Cannot INSERT into a compressed external table from multiple tables, , Decompression failed: ERROR: Wrong header (size)

- The **nz.fq.splits** property sets the number of concurrent JDBC load sessions to the NPS host.

```
<property>
  <name>nz.fq.splits</name>
  <value>12</value>
</property>
```

- The **fq.custom.exttab.options** is an optional parameter for advanced users. It allows to provide user defined options to external tables. All possible external table options and values are listed in the IBM PureData System for Analytics Knowledge Center at [https://www.ibm.com/support/knowledgecenter/en/SSULQD\\_7.2.1/com.ibm.nz.load.doc/c\\_load\\_options.html](https://www.ibm.com/support/knowledgecenter/en/SSULQD_7.2.1/com.ibm.nz.load.doc/c_load_options.html). You can provide multiple pairs of external table option and its value, separated by a space.

**Note:** The options provided with the **fq.custom.exttab.options** parameter overwrite the corresponding ones existing in IBM Fluid Query code.

```
<property>
  <name>fq.custom.exttab.options</name>
  <value>MaxErrors 1 SocketBufSize 8000000</value>
</property>
```

- The **nz.fq.break.on.error** property, when set to true, stops import execution in case of error during the import of more than one table or whole database. When the import is finished, information about transferred tables is shown.

```
<property>
  <name>nz.fq.break.on.error</name>
  <value>>false</value>
</property>
```

- The **nz.fq.append.mode** property specifies how data is transferred to the specified HDFS location. The property takes the following values:
  - **create** During import, new table is created in the target database. If the table with the same name already exists, import fails.
  - **overwrite** During import, a table from the source database is imported into the target database, and if any table with the same name already existed in the target database, it is replaced (overwritten).
  - **append** The data from the source table is imported into the target table and appended to the already existing records. That is, if a table with the same name already exists in the target database, all data from the source table is added (appended) to the existing table.
  - **incremental** Incremental mode is only available for import, not for export. During import, only the data that was added to the source table since the last successful import operation is appended to the existing target table.

For more information on append modes, read “Append modes for import and export” on page 4-23.

Default value for this property is create.

```
<property>
  <name>nz.fq.append.mode</name>
  <value>overwrite</value>
</property>
```

- The **nz.fq.ignore.deleted** parameter is valid only with **nz.fq.append.mode** set to **incremental**. If set to false, it stops the incremental import to Hadoop when there was an update or deletion in the table on NPS after the last import, as it leads to data inconsistency.

```
<property>
  <name>nz.fq.ignore.deleted</name>
  <value>>false</value>
</property>
```

- The **nz.fq.sql.metadata** property defines whether IBM Fluid Query will create the table in Hive during import. By default, it is set to **true**.

```
<property>
  <name>nz.fq.sql.metadata</name>
  <value>>true</value>
</property>
```

If you set the property to **false**, the table will not be created in Hive and IBM Fluid Query will only import data files and put them on HDFS.

## Hive options

- The **fq.hive.usevarchar** property determines the rules of conversion for VARCHAR column. VARCHAR type is supported by Hive version 0.12 or later. If the property is set to true then data type is VARCHAR after import. If false, the data type is converted to STRING. Default value is false because there is a number of limitations when VARCHAR is used, as described in “Limitations on VARCHAR data types” on page 5-11.

```
<property>
  <name>nz.fq.hive.usevarchar</name>
  <value>>false</value>
</property>
```

- The **nz.fq.hive.usedate** property determines the rules of conversion for NPS DATE column. For Hive versions 0.12 and above, the NPS DATE can be mapped to Hive DATE instead of STRING when the parameter is set to true. By default the value is set to false.

```
<property>
  <name>nz.fq.hive.usedate</name>
  <value>>false</value>
</property>
```

- The **fq.hive.clustered.by** parameter lists table column(s) which will be used for clustering. To enable default clustering, keep this property empty and set **fq.hive.cluster.buckets**.

```
<property>
  <name>fq.hive.clustered.by</name>
  <value></value>
</property>
```

- The **fq.hive.clustered.buckets** parameter provides an integer value which determines the amount of buckets that can be created during clustering. This property must be set to enable clustering.

```
<property>
  <name>fq.hive.clustered.buckets</name>
  <value></value>
</property>
```

- The **fq.hive.partitioned.by** parameter lists columns in a table that will be used for partitioning.

```
<property>
  <name>fq.hive.partitioned.by</name>
  <value></value>
</property>
```

## Hadoop JDBC connection (remote connection) properties

- The **nz.fq.sql.server** property sets the Hive server address on Hadoop where the imported table is created.

```
<property>
  <name>nz.fq.sql.server</name>
  <value>rack1-master</value>
</property>
```

- The **nz.fq.sql.port** property sets the Hive server port number on Hadoop.

```
<property>
  <name>nz.fq.sql.port</name>
  <value>10000</value>
</property>
```

- The **nz.fq.sql.type** property sets the server type. Supported types are **hive2** or **bigsql**.

```
<property>
  <name>nz.fq.sql.type</name>
  <value>hive2</value>
</property>
```

**Note:** Use this property for remote import, not export.

## JDBC authentication properties

**Note:** You must provide values for either user and password or for the Kerberos service principal name.

- The **nz.fq.sql.user** property sets the user name. It is required if you want to use a User/Password authentication.

```
<property>
  <name>nz.fq.sql.user</name>
  <value>biadmin</value>
</property>
```

- The **nz.fq.sql.password** property sets the password. It is required if user name was provided.

```
<property>
  <name>nz.fq.sql.password</name>
  <value>passw0rd</value>
</property>
```

## JDBC SSL properties

- The **nz.fq.sql.ssl** property defines whether SSL is required to connect to the selected Hadoop SQL server. Value can be **true** or **false**.

```
<property>
  <name>nz.fq.sql.ssl</name>
  <value>>false</value>
</property>
```

- The **nz.fq.sql.sslTrustStore** property sets the path to the SSL trustStore that is to be used.

```
<property>
  <name>nz.fq.sql.sslTrustStore</name>
  <value>${HIVE_HOME}/src/data/files/cacerts_test.jks</value>
</property>
```

- The **nz.fq.sql.sslTrustStorePassword** property sets the password to the specified SSL trustStore.

```
<property>
  <name>nz.fq.sql.sslTrustStorePassword</name>
  <value>passw0rd</value>
</property>
```

## Hadoop remote execution settings

- The **nz.fq.hadoop.remote.user** parameter is used when running data movement from a system different than Hadoop. It specifies the remote Hadoop user name that will run data movement. This user should have access to HDFS.

```
<property>
  <name>nz.fq.hadoop.remote.user</name>
  <value></value>
</property>
```

- The **nz.fq.hadoop.client.configs** parameter is used when running data movement from a system different than Hadoop. It specifies the path to the downloaded client configuration files.

```
<property>
  <name>nz.fq.hadoop.client.configs</name>
  <value></value>
</property>
```

## JDBC Kerberos properties

- The **nz.fq.kerberos.principal** property sets the Kerberos principal name. Required if Kerberos authentication should be used.

```
<property>
  <name>nz.fq.kerberos.principal</name>
  <value>hdfs/rack1-master.hadoop.ibm.com@REALM.IBM.COM</value>
</property>
```

- The **nz.fq.kerberos.keytab** property sets the Kerberos keytab. Required if Kerberos authentication should be used.

```
<property>
  <name>nz.fq.kerberos.keytab</name>
  <value>/path/to/your/client.keytab</value>
</property>
```

- The **nz.fq.sql.spn** property sets the Kerberos service principal name. It is required if you want to use Kerberos authentication.

```
<property>
  <name>nz.fq.sql.spn</name>
  <value>hive/horton-master.ibm.com@XXXXXX.IBM.COM</value>
</property>
```

## Additional settings

**Note:** When using BigSQL, you might need to add the following property to the `fq-import-remote-conf.xml` file:

```
<property>
  <name>nz.fq.sql.bigsql.v3compatibility</name>
  <value>true</value>
</property>
```

The property solves the problem with mapping a NUMERIC field, as described in “Limitations on decimal values in BigInsights” on page 5-7.

## Data export configuration XML

Learn how to customize the XML file which configures exporting data from Hadoop to NPS.

### General configuration

- The **nz.fq.command** property sets the type of data movement: import (NPS->Hadoop) or export (Hadoop->NPS).

```
<property>
  <name>nz.fq.command</name>
  <value>export</value>
</property>
```

### HDFS information

- The **nz.fq.input.path** parameter sets the directory on HDFS where the retrieved data is stored.

```
<property>
  <name>nz.fq.input.path</name>
  <value>/nzbackup/fqtest1</value>
</property>
```

**Note:** The properties **nz.fq.input.path** and **nz.fq.hive.tablename** are exclusive, that is only one of them might be used to define the location of the data that you want to export. **nz.fq.input.path** provides more flexibility, as **nz.fq.hive.tablename** allows you to specify one table only, not multiple tables.

- The **nz.fq.format fielddelim** parameter sets the integer value of the single character field delimiter in the plain text output file.

```
<property>
  <name>nz.fq.format.fielddelim</name>
  <value>124</value>
</property>
```

- The **nz.fq.format.null** property defines how the NULL value will be represented. Default is NULL. The value cannot be longer than 4 characters.

```
<property>
  <name>nz.fq.format.null</name>
  <value>NULL</value>
</property>
```

- The **nz.fq.fs.temp** parameter sets the location of temporary files (such as logs and status files) on HDFS.

```
<property>
  <name>nz.fq.fs.temp</name>
  <value>/tmp</value>
</property>
```

### NPS information

- The **nz.fq.nps.db** property sets the NPS database name. Include double quotations around delimited database names.



```
<property>
  <name>nz.fq.nps.db</name>
  <value>dev</value>
</property>
```

- The **nz.fq.table** parameter can be used to rename a specific table during export so that the exported table on NPS has a different name than the source table on Hadoop. Include double quotations around delimited table names. The format of this value is **<SCHEMA>.<table>**.

```
<property>
  <name>nz.fq.table</name>
  <value>ADMIN.tab</value>
</property>
```

**Restriction:** If full schema support is enabled on your NPS system, make sure to provide the schema name together with the table name in this property.

- The **nz.fq.tables** property provides a list of target tables for the exported data, but it also allows to filter these tables with the use of a wildcard character \*. The use of the \* filter is described in detail in “Importing or exporting multiple tables at a time” on page 4-35. You must include double quotations around delimited table names.
- The **nz.fq.append.mode** property defines how data is transferred to the specified HDFS location. The property takes the following values:
  - **create** During import, new table is created in the target database. If the table with the same name already exists, import fails.
  - **overwrite** During import, a table from the source database is imported into the target database, and if any table with the same name already existed in the target database, it is replaced (overwritten).
  - **append** The data from the source table is imported into the target table and appended to the already existing records. That is, if a table with the same name already exists in the target database, all data from the source table is added (appended) to the existing table.

For more information on append modes, read “Append modes for import and export” on page 4-23.

Default value for this property is create.

```
<property>
  <name>nz.fq.append.mode</name>
  <value>create</value>
</property>
```

- The **nz.fq.nps.server** property sets the wall IP address or the fully qualified host name of the NPS server.

```
<property>
  <name>nz.fq.nps.server</name>
  <value>hostname.ibm.com</value>
</property>
```

- The **nz.fq.nps.port** property sets the port number for the NPS database instance `NZ_DBMS_PORT`.

```
<property>
  <name>nz.fq.nps.port</name>
  <value>5480</value>
</property>
```

- The **nz.fq.nps.user** property sets The NPS database user account name for access to the database.

```
<property>
  <name>nz.fq.nps.user</name>
  <value>admin</value>
</property>
```

- The **nz.fq.nps.password** property sets the password for the NPS database user account.

```
<property>
  <name>nz.fq.nps.password</name>
  <value>password</value>
</property>
```

- The **nz.fq.nps.ssl** property sets the NPS server connection type. When set to true, then onlySecured JDBC mode is used. Default is false.

```
<property>
  <name>nz.fq.nps.ssl</name>
  <value>false</value>
</property>
```

- The **nz.fq.nps.ssl.cacertificate** property sets the full path to the CA Certificate file that is stored on HDFS and used to authenticate connections. Used only when the SSL flag is true. If not provided, then all connections are accepted.

```
<property>
  <name>nz.fq.nps.ssl.cacertificate</name>
  <value></value>
</property>
```

- The **nz.fq.exttab.columns** parameter sets the external table column names in proper order, for example CODE, TITLE, PRICE. The default value is \*. Detailed syntax is described in "Transient External Tables" in *IBM Netezza Data Loading Guide*.

```
<property>
  <name>nz.fq.exttab.columns</name>
  <value>*</value>
</property>
```

- The **nz.fq.exttab.schema** parameter sets the external table schema definition, for example CODE CHAR(5), TITLE VARCHAR(255), PRICE INTEGER.

```
<property>
  <name>nz.fq.exttab.schema</name>
  <value></value>
</property>
```

The default value is the empty string, which uses the schema of the target table. Detailed syntax for transient external tables is described in the *IBM Netezza Data Loading Guide*.

- The **fq.custom.exttab.options** is an optional parameter for advanced users. It allows to provide user defined options to external tables. All possible external table options and values are listed in the IBM PureData System for Analytics Knowledge Center at [https://www.ibm.com/support/knowledgecenter/en/SSULQD\\_7.2.1/com.ibm.nz.load.doc/c\\_load\\_options.html](https://www.ibm.com/support/knowledgecenter/en/SSULQD_7.2.1/com.ibm.nz.load.doc/c_load_options.html). You can provide multiple pairs of external table option and its value, separated by a space.

**Note:** The options provided with the **fq.custom.exttab.options** parameter overwrite the corresponding ones existing in IBM Fluid Query code.

```
<property>
  <name>fq.custom.exttab.options</name>
  <value>MaxErrors 1 SocketBufSize 8000000</value>
</property>
```

- The **nz.fq.splits** property sets the number of concurrent JDBC load sessions to the NPS host.

```
<property>
  <name>nz.fq.splits</name>
  <value>12</value>
</property>
```

## Checksum options

- The **fq.checksum** parameter indicates whether the checksum has to be calculated after data movement. Available values:
  - NONE - no checksum (default)
  - FULL - calculate checksum using all columns in each table
  - ROWCOUNT - only row count
  - COLUMNS - calculate checksum using columns specified in **fq.checksum.columns**.

```
<property>
  <name>fq.checksum</name>
  <value>NONE</value>
</property>
```

- The **fq.checksum.columns** parameter provides a list of columns for checksum calculation. When used with multiple tables import or export, checksum is calculated only on the listed columns within the tables. If no matching columns are found, only row count check is performed.

```
<property>
  <name>fq.checksum.columns</name>
  <value></value>
</property>
```

## Hadoop Hive SQL connection configuration

### JDBC connection properties

- The **nz.fq.sql.server** property sets the Hive server address on Hadoop where the imported table is created.

```
<property>
  <name>nz.fq.sql.server</name>
  <value>rack1-master</value>
</property>
```

- The **nz.fq.sql.port** property sets the Hive server port number on Hadoop.

```
<property>
  <name>nz.fq.sql.port</name>
  <value>10000</value>
</property>
```

- Import only: The **nz.fq.sql.type** property sets the server type. Supported types are **hive2** or **bigsql**.

```
<property>
  <name>nz.fq.sql.type</name>
  <value>hive2</value>
</property>
```

**Note:** Use this property for remote import, not export.

## JDBC authentication properties

**Note:** You must provide values for either user and password or for the Kerberos service principal name.

- The **nz.fq.sql.user** property sets the user name. It is required if you want to use a User/Password authentication.

```
<property>
  <name>nz.fq.sql.user</name>
  <value>biadmin</value>
</property>
```

- The **nz.fq.sql.password** property sets the password. It is required if user name was provided.

```
<property>
  <name>nz.fq.sql.password</name>
  <value>passw0rd</value>
</property>
```

## JDBC SSL properties

- The **nz.fq.sql.ssl** property defines whether SSL is required to connect to the selected Hadoop SQL server. Value can be **true** or **false**.

```
<property>
  <name>nz.fq.sql.ssl</name>
  <value>>false</value>
</property>
```

- The **nz.fq.sql.sslTrustStore** property sets the path to the SSL trustStore that is to be used.

```
<property>
  <name>nz.fq.sql.sslTrustStore</name>
  <value>$HIVE_HOME/src/data/files/cacerts_test.jks</value>
</property>
```

- The **nz.fq.sql.sslTrustStorePassword** property sets the password to the specified SSL trustStore.

```
<property>
  <name>nz.fq.sql.sslTrustStorePassword</name>
  <value>passw0rd</value>
</property>
```

## JDBC Kerberos properties

- The **nz.fq.kerberos.principal** property sets the Kerberos principal name. Required if Kerberos authentication should be used.

```
<property>
  <name>nz.fq.kerberos.principal</name>
  <value>hdfs/rack1-master.hadoop.ibm.com@REALM.IBM.COM</value>
</property>
```

- The **nz.fq.kerberos.keytab** property sets the Kerberos keytab. Required if Kerberos authentication should be used.

```
<property>
  <name>nz.fq.kerberos.keytab</name>
  <value>/path/to/your/client.keytab</value>
</property>
```

- The **nz.fq.sql.spn** property sets the Kerberos service principal name. It is required if you want to use Kerberos authentication.

```
<property>
  <name>nz.fq.sql.spn</name>
  <value>hive/orton-master.ibm.com@XXXXXX.IBM.COM</value>
</property>
```

## Hadoop remote execution settings

- The **nz.fq.hadoop.remote.user** parameter is used when running data movement from a system different than Hadoop. It specifies the remote Hadoop user name that will run data movement. This user should have access to HDFS.

```
<property>
  <name>nz.fq.hadoop.remote.user</name>
  <value></value>
</property>
```

- The **nz.fq.hadoop.client.configs** parameter is used when running data movement from a system different than Hadoop. It specifies the path to the downloaded client configuration files.

```
<property>
  <name>nz.fq.hadoop.client.configs</name>
  <value></value>
</property>
```

## Additional settings

- The **nz.fq.hive.tablename** parameter provides the name of the source table on Hive. When this value is provided, the **nz.fq.input.path** must be empty.

```
<property>
  <name>nz.fq.hive.tablename</name>
  <value></value>
</property>
```

- The **nz.fq.hive.set.distribute.on** parameter is used to specify data distribution key by column name(s). Can be used in case of Hive table export. Empty string is default, which means that data will be distributed randomly.

```
<property>
  <name>nz.fq.hive.set.distribute.on</name>
  <value></value>
</property>
```

- The **fq.hive.where** parameter can be used to define an SQL WHERE clause to use for selecting the data to transfer during export of a Hive table. This parameter only works when **fq.hive.tablename** is set.
- The **nz.fq.skip.table.structure.validation** parameter is used in case of exporting a Hive table to the already existing destination table in warehouse. When this value is set to true, the validation between the structures of source Hive table and destination warehouse table is omitted. Default value is false.

```
<property>
  <name>nz.fq.skip.table.structure.validation</name>
  <value>>false</value>
</property>
```

## Preparing the configuration XML file for the Hadoop file formats

IBM Fluid Query supports conversion to the following Hadoop-specific file formats: Parquet, Avro, ORC, RCFile. You can import the files from NPS and store them on Hadoop in a format of your choice.

### Procedure

1. Edit the `fq-import-remote-conf.xml` template.
2. Define the JDBC connection properties as described in Remote import and export configuration XML.
3. Set the **nz.fq.data.format** property using one of the options: PARQUET, ORC, RCFILE, AVRO

```
<property>
  <name>nz.fq.data.format</name>
  <value>PARQUET</value>
</property>
```

4. Set the **nz.fq.output.compressed** parameter to select compression type. If you set the property to false or leave it empty, the default compression type that is specified on Hadoop for the selected format will be used. Depending on the format that you use, select one of the following values:
  - PARQUET: Snappy, gzip, uncompressed
  - ORC: NONE, ZLIB, SNAPPY
  - RCFILE: The value has to contain the exact class name of the codec which is available on Hadoop system. For example:  
`org.apache.hadoop.io.compress.SnappyCodec`
  - AVRO: snappy, deflate
5. Because mixed mode of transfer is not supported when using Hadoop formats, the **nz.fq.compress** property must be set to false or empty.
6. Save the XML file and take note of the file path.

## Results

You can now run the import task as described in “Configuring and running data movement” on page 4-27.

### Data format properties combinations

Learn how to use parameters for format and compression to define the output format for data movement, especially when transferring to the Hadoop formats.

### Setting the properties to achieve the selected format on Hadoop

Following is a list of format and compression settings that you use in the configuration XML, organized by the output format.

- Hadoop formats:
  - **nz.fq.data.format** set to one of orc/parquet/avro/rcfile
  - **nz.fq.compress** keep empty
  - **nz.fq.output.compressed** (optional) set to compression supported by a particular format
- nzbak (Netezza compressed mode):
  - **nz.fq.data.format** set to nzbak
  - **nz.fq.compress** keep empty
  - **nz.fq.output.compressed** keep empty
- Text mode:
  - **nz.fq.data.format** set to txt
  - **nz.fq.compress** set to false
  - **nz.fq.output.compressed** keep empty
- Mixed mode:
  - **nz.fq.data.format** set to txt
  - **nz.fq.compress** set to true
  - **nz.fq.output.compressed** keep empty
- Legacy text/mixed/nzbak specification: you can still use **nz.fq.compress=true/false** and **nz.fq.output.compressed=true/false**, but then the **nz.fq.data.format** property must be deleted or empty.

The following sections list all possible combinations of the mentioned parameters.

### Text, mixed and compressed modes as in versions prior to 1.7

In IBM Fluid Query versions prior to 1.7, three modes of import were available: text, mixed and compressed. You can still use the same properties in 1.7, when you delete the **nz.fq.data.format** property or leave it empty.

Table 4-1. Format and compression parameters for previous versions of Fluid Query

<b>nz.fq.data.format</b>	<b>nz.fq.compress</b>	<b>nz.fq.output.compressed</b>	<b>Result</b>
empty or not existing	TRUE	TRUE	Compressed (Netezza)
empty or not existing	FALSE	FALSE	Text mode
empty or not existing	TRUE	FALSE	Mixed mode
empty or not existing	FALSE	TRUE	Error message
empty or not existing	empty or not existing	empty or not existing	Text mode

## The use of `nz.fq.data.format` in IBM Fluid Query 1.7

You can use the `nz.fq.data.format` parameter in combination with the compression parameters to either use the previous modes of import, or to use the Hadoop-specific formats.

Table 4-2. Format and compression parameters for IBM Fluid Query 1.7

<code>nz.fq.data.format</code>	<code>nz.fq.compress</code>	<code>nz.fq.output.compressed</code>	<code>nz.fq.splits</code>	Result
TXT	FALSE	FALSE	n/a	Text
	TRUE	FALSE	n/a	Mixed
	TRUE	TRUE	n/a	Error message
	FALSE	TRUE	n/a	Error message
	empty or not existing	empty or not existing	n/a	Text
	any	TRUE	n/a	Error message
NZBAK	empty or not existing	empty or not existing	n/a	Compressed (Netezza) (empty or not existing = default TRUE)
	TRUE	empty or not existing	n/a	Compressed (Netezza) (empty or not existing = default TRUE)
	empty or not existing	TRUE	n/a	Compressed (Netezza) (empty or not existing = default TRUE)
	TRUE	TRUE	n/a	Compressed (Netezza) (empty or not existing = default TRUE)
	FALSE	any	n/a	Error message: <code>nz.fq.compress</code> set to FALSE is not allowed for NZBAK
	any	FALSE	n/a	Error message: <code>nz.fq.output.compressed</code> set to FALSE is not allowed for NZBAK
	FALSE	FALSE	n/a	Error message: <code>nz.fq.compress</code> and <code>nz.fq.output.compressed</code> set to FALSE is not allowed for NZBAK)



Table 4-2. Format and compression parameters for IBM Fluid Query 1.7 (continued)

<code>nz.fq.data.format</code>	<code>nz.fq.compress</code>	<code>nz.fq.output.compressed</code>	<code>nz.fq.splits</code>	Result
One of the following values:  PARQUET ORC RCFILE AVRO	TRUE	any	n/a	Error message ( <b>nz.fq.compress</b> set to TRUE is not allowed for new formats - mixed mode is not supported).
	any	TRUE	n/a	Error message ( <b>nz.fq.output.compressed</b> set to TRUE is not allowed for Hadoop formats. The value of this parameter depends on the used format).
	FALSE or empty	Any of the listed values, depending on the format that you use:  PARQUET: Snappy, gzip, uncompressed  ORC: NONE, ZLIB, SNAPPY  RCFILE: The value has to contain the exact class name of the codec which is available on Hadoop system. For example: org.apache.hadoop.io.compress.SnappyCodec  AVRO: snappy, deflate	any value from 1 to 30	Conversion to the selected Hadoop format with the selected compression type
	FALSE or empty	FALSE or empty	any value from 1 to 30	Conversion to a selected Hadoop format with any default compression that is specified on Hadoop for the selected format.
	any	any	>30	Error message ( <b>nz.fq.splits</b> > 30 is not allowed for Hadoop formats)

## Append modes for import and export

You use **nz.fq.append.mode** property in the configuration XML file to define how and if the records are transferred to the specified target database.

Property **nz.fq.append.mode** defines the way in which new data is added to the already existing data. The property can be used with both, import and export operations.

The **nz.fq.append.mode** property takes the following values:

**create** During import, new table is created in the target database. If the table with the same name already exists, import fails. This value is set as default.

### overwrite

During import, a table from the source database is imported into the target database, and if any table with the same name already existed in the target database, it is replaced (overwritten).

**append** The data from the source table is imported into the target table and appended to the already existing records. That is, if a table with the same name already exists in the target database, all data from the source table is added (appended) to the existing table. Note that this might cause duplicate entries in the target table. If the table does not exist in the target database, it is created.

### incremental

Incremental mode is only available for import, not for export. During import, only the data that was added to the source table since the last successful import operation is appended to the existing target table. In incremental mode data consistency is checked based on transactions performed on NPS. If any deletions or updates were made in the table on NPS after the last import, these changes cannot be propagated to Hadoop. To prevent data inconsistency, import stops and the following error message is displayed:

Error message: Some records have been deleted or updated after last import. Full import is required to have consistent data. To run incremental import anyway please set `nz.fq.ignore.deleted` to true.

You can then use the **overwrite** mode to replace the table on Hadoop with the updated table on NPS, or use **append** mode to add all data from the NPS table to the existing Hadoop table. Note that this might result in duplicate entries.

You can also use the `nz.fq.ignore.deleted` flag to ignore the error message and import data in **incremental** mode anyway. Note that data you have in Hadoop table will differ from data in NPS table.

### Example:

In the following example, there is one table on NPS and a corresponding one on Hadoop. Table on NPS was updated with one new row.

Table 4-3. Table on NPS

col1	col2
1	1
2	2
3	3

Table 4-4. Table on Hadoop

col1	col2
1	1
2	2

When you import the table from NPS to Hadoop using `nz.fq.append.mode = append` then the resulting table on Hadoop will look as follows:

Table 4-5. Table on Hadoop after import in append mode

col1	col2
1	1
2	2
1	1
2	2
3	3

When you import the table from NPS to Hadoop but you use `nz.fq.append.mode = incremental` then the resulting table on Hadoop will look as follows:

Table 4-6. Table on Hadoop after incremental import

col1	col2
1	1
2	2
3	3

## Using checksum to check data consistency

The checksum functionality introduced in IBM Fluid Query version 1.7.1 allows you to verify if, after data movement, the data is consistent and no unintended changes were introduced during the process of migration.

### About this task

You can set checksum calculation for import or export using the `fq.checksum` and `fq.checksum.columns` properties in the XML configuration files. By default, the functionality is disabled.

Depending on the size of the data that you plan to move, checksum calculation might slow down the process of data movement. Therefore, there are three modes of checksum calculation:

- FULL, where all columns in all tables are used for checksum calculation;
- COLUMNS, where you can specify which columns are to be used for calculation;
- ROWCOUNT, where only the rows are counted.

To enable checksum calculation, follow the steps described in the procedure.

### Procedure

1. When configuring import or export, set the **nz.fq.checksum** parameter in the configuration XML file. By default, the value is NONE, so the functionality is disabled. Choose one of the following values:
  - FULL
  - COLUMNS
  - ROWCOUNT

```
<property>
  <name>nz.fq.checksum</name>
  <value>COLUMNS</value>
</property>
```

2. If you set **nz.fq.checksum** to COLUMNS, you must specify the list of columns in another parameter **nz.fq.checksum.columns**. The checksum is then counted on all columns with the listed names, even if the same column exists in more than one table.

```
<property>
  <name>nz.fq.checksum.columns</name>
  <value>COL1,COL3,COL5</value>
</property>
```

3. Proceed with configuration and run data movement as usual.

### Results

- When running import, checksum is calculated on NPS before import, and on Hive or BigSQL after import. The values of checksums are compared as a

validation. Note that checksum is not calculated if you decide not to create Hive/BigSQL metadata (nz.fq.sql.metadata = false).

- When running export, checksum is calculated on Hive/BigSQL before export and on NPS after export. The values of checksums are compared as a validation.

Sample output:

```
2017-02-23 16:53:43,362 52369 [main] INFO com.ibm.nz.fq.NzTransfer - Import summary:
Used filter: checksum_empty_test
Found 1 tables matching the filter: CHECKSUM_EMPTY_TEST
Imported 1 out of 1 table(s): CHECKSUM_EMPTY_TEST

=====
===== CHECKSUM REPORT =====
=====
      TABLE      STATUS  PDA CNT    HD CNT          PDA SUM          HD SUM
ADMIN.CHECKSUM_EMPTY_TEST  EQUAL         1         1    -2.1239802369048    -2.1239802369048
=====

2017-02-23 16:53:43,362 52369 [main] DEBUG com.ibm.nz.fq.NzTransfer - PERF: Fdm done | Total execution time: 5
Done
2017-02-23 16:53:43,362 52369 [main] INFO com.ibm.nz.fq.NzTransfer - Exiting with code: 0
```

The report contains the following information for every table:

#### TABLE

Table name

#### STATUS

One of the following statuses:

- EQUAL - checksums are equal
- ERR:CNT - the number of rows is different
- ERR:SUM - the checksums are different
- ERROR - other error

#### PDA CNT

Row count on NPS

#### HD CNT

Row count on Hadoop

#### PDA SUM

Checksum on NPS

#### HD SUM

Checksum on Hadoop

Sample output for checksums that are not equal:

```
=====
===== CHECKSUM REPORT =====
=====
The checksum found 1 problems:
      TABLE      STATUS  PDA CNT    HD CNT          PDA SUM          HD SUM
ADMIN.CHECKSUM_TEST  ERR:SUM         47         47    26.7232625946026    17.0031239082296
=====
```

#### Note:

- There might be a discrepancy between the checksum calculated on NPS and Hive when both of the mentioned conditions are met:

1. On NPS table, within the VARCHAR column there is at least one value starting with character which belongs to Latin-9 and does not belong to ASCII (that means, the code of this character is above 128 and below 255).
2. The Hive table has default `serialization.encoding ( UTF-8 )`

Hive has a problem with processing characters with code above 128 in string columns. To solve this problem, you can change table encoding to ISO-8859-15 (that is, Latin-9):

```
alter table <tabName> set SERDEPROPERTIES
('serialization.encoding'='ISO-8859-15');
```

Note that this can only be done when there are no NVARCHAR columns in the data. For more information, see <https://community.hortonworks.com/articles/58548/processing-files-in-hive-using-native-non-utf8-cha.html>

- Checksum calculation fails if the column names are different in source and target table.

## Encrypting the configuration XML files

The passwords and information stored in the import and export configuration XML files are in plain text and thus visible to users who can log in to the NPS system or the Hadoop nodes. After you edit and save the configuration file, you can encrypt the file to protect the information it contains.

### About this task

To encrypt the file, you use the `nzcodec.jar` file with special command arguments.

### Procedure

- On a Hadoop node where the data movement feature is installed, you can encrypt an XML configuration file using a command similar to the following:  

```
hadoop jar /fluidqueryLocal/nzcodec.jar -encryptConfig -conf /tmp/test_password.xml
```

FluidQuery version: 1.5.0.0 [Build 150626-182]  
Encrypting configuration file.  
Saving configuration file.  
Encryption done.

- On an NPS system, you can use a command similar to the following:  

```
/nz/export/ae/products/fluidquery/ibm-java-i386-60/jre/bin/java
-cp '/nz/export/ae/products/fluidquery/libs/nzcodec.jar:path/to/hadoop/drivers/'
com.ibm.nz.fq.NzTransfer -encryptConfig -conf /file_to_encrypt.xml
```

Following is a sample command for an import XML configuration file:

```
/nz/export/ae/products/fluidquery/ibm-java-i386-60/jre/bin/java -cp
'/nz/export/ae/products/fluidquery/libs/nzcodec.jar:/nz/export/ae/products/fluidquery/libs/ibm/fqdm/'
com.ibm.nz.fq.NzTransfer -encryptConfig
-conf /nz/export/ae/products/fluidquery/conf/fq-remote-conf.xml
```

---

## Configuring and running data movement

After you have prepared the XML configuration file, you can use IBM Fluid Query to transfer data between NPS and Hadoop. You can run the data transfer from the Hadoop system, from the NPS system, or from any other systems that run Java.

### Before you begin

The data movement feature transfers data directly between Hadoop data nodes and NPS. Therefore, a working network connection between all data nodes and NPS is a prerequisite.

## Validating data movement configuration

It is recommended that you validate your connection configuration before running data movement.

### About this task

Whether you run data movement from Hadoop or from NPS, you can use a validation process to verify that configuration details are correct. The script performs the following checks:

- Access to HDFS
- Connection to warehouse
- Connection to database engine (Hive or BigSQL)
- Map-reduce jobs

It also checks the completeness of the configuration XML file, its read access rights, and whether the values provided are allowed.

### Procedure

- On Hadoop, run the following command before you run data movement:  
`hadoop jar /fluidqueryLocal/nzcodec.jar -conf <configuration_file>.xml -validate`
- On NPS the validation is performed by default when running the **fqConfigure.sh** script. You can skip this process by adding `--no-validation` parameter when running the command.

### Results

Status is displayed on screen. A complete validation log can be found in the following log directories:

- on Hadoop in `/fluidqueryLocal/var/log`
- on NPS in `/nz/export/ae/products/fluidquery/logs`

## Running data movement from Hadoop

You can start both import and export operations by running the `fdm.sh` script from Hadoop.

### Before you begin

Ensure that you have installed the IBM Fluid Query software on Hadoop, as described in “Installing or upgrading the data movement feature” on page 2-8.

During installation, you are asked to provide the paths to Hadoop `/lib` and `/conf` files. These details are then used by `fdm.sh` script when running data movement.

Once you have installed the Fluid Query package on Hadoop, you can find `fdm.sh` in `/fluidqueryLocal/`.

### Procedure

1. Depending on whether you are running an import or an export transfer, prepare the XML configuration file using one of the available templates, as described in “Preparing configuration XML files” on page 4-4. Each file defines a single transfer job, that is single or multiple tables from or to a particular NPS database.

2. Optionally, you can validate connection configuration as described in “Validating data movement configuration” on page 4-28.
3. From the command line, run the `fdm.sh` and provide the import or export XML configuration file as the `-conf` parameter. For example:

```
fdm.sh -conf fq-import-conf.xml
```

**Tip:** You can also overwrite any parameters included in the configuration file with `-D` parameter:

```
fdm.sh -conf fq-import-conf.xml -D nz.fq.tables=import_test -D  
nz.fq.append.mode=overwrite
```

## Results

Data transfer is performed based on the properties that you provided.

## Configuring connection from Hadoop to NPS with Kerberos authentication

If you want to perform import and export operations from Hadoop using Kerberos authentication, you must first perform some manual configuration steps.

### Before you begin

If your environment has multiple Hadoop data nodes, connecting to NPS with Kerberos authentication might be a time-consuming task, as it requires manual steps performed on each data node separately. An alternative solution is to define a user on NPS with local authentication (user/password), dedicated for data movement tasks, and then execute data movement as this user.

**Note:** IBM Fluid Query does not support Kerberized IBM BigInsights 4.2.

### Procedure

1. Create the `login.config` file on all Hadoop nodes. The file must be in a location that is accessible to the user who runs the MapReduce jobs, such as YARN, on the data nodes.

```
touch login.config
```

2. Put the following text in the `login.config` file:

- For Hortonworks and Cloudera (Oracle Java, OpenJDK):

```
EntryModuleName{  
    com.sun.security.auth.module.Krb5LoginModule required  
    useTicketCache = true  
    debug = true;  
};
```

- For BigInsights (IBM Java):

```
EntryModuleName{  
    com.ibm.security.auth.module.Krb5LoginModule required  
    useTicketCache = true  
    debug = true;  
};
```

### Note:

If you need to run data movement from Hadoop without Kerberos to NPS with Kerberos, the `EntryModuleName` at `login.config` must contain principal and keytab for data movement to work:

```
EntryModuleName{
    com.sun.security.auth.module.Krb5LoginModule required
    debug = true
    useTicketCache = true
    principal = kf_keytab
    useKeyTab = true
    keyTab = "/nzscratch/BVT/fluidquery/kf_keytab_vmKerberos.keytab";
};
```

Note that with such configuration, every application on Hadoop that has access to the `login.config` also has access to NPS with this principal.

3. Edit the `java.security` file so that Hadoop JVM can locate the `login.config` file, for example:
  - a. Edit: `/opt/ibm/biginsights/jdk/jre/lib/security/java.security`
  - b. Add: **`login.config.url.1=file:/home/biadmin/login.config`**

**Note:** The file paths in this example are valid for BigInsights 3.

4. Edit the `/etc/krb5.conf` file so that it reflects the actual Kerberos setup. You can copy this file from NPS.

### What to do next

Run these steps to check whether the Kerberos authentication is configured correctly:

1. On each data node authenticate on Kerberos using **`kinit`**, for example:

```
kinit kerberos_example -kt /tmp/kerberos_example.keytab
```

2. Copy the `nzjdbc3.jar` file to `/fluidqueryLocal` on the Hadoop node.
3. Run the following command on Hadoop:

```
/usr/bin/java -Djava.security.krb5.conf=/etc/krb5.conf
-jar /fluidqueryLocal/nzjdbc3.jar -t
```

The output of this command should be similar to the following:

```
Success
NPS Product Version: Release 7.2.0.0 [Build 40845]
Netezza JDBC Driver Version: Release 7.2.0.0 driver [build 40845]
```

The master node and all the data nodes in the Hadoop cluster must be able to run this command correctly. The Java that is used (`/usr/bin/java`) should be the Hadoop JVM Java, the same one that is used by Hadoop when running the **`hadoop jar`** command.

### Deprecated: Running data movement from Hadoop with `nzcodec.jar`

**Note:** The following method of running data movement from Hadoop is deprecated. Starting with IBM Fluid Query version 1.7.1 you can run data



movement from Hadoop using the `fdm.sh` script, which is described in “Running data movement from Hadoop” on page 4-28.

You can run the data transfer with `nzcodec.jar` directly from your Hadoop nodes on which you have installed the data movement software.

## Before you begin

Ensure that you have installed the IBM Fluid Query software on Hadoop, as described in “Installing or upgrading the data movement feature” on page 2-8.

## Procedure

1. Depending on whether you are running an import or an export transfer, prepare the XML configuration file on the basis of one of the templates:

- `fq-import-conf.xml`
- `fq-import-remote-conf.xml`
- `fq-export-conf.xml`
- `fq-export-remote-conf.xml`

Each file defines a single transfer job, that is single or multiple tables from or to a particular NPS database. The XML configuration details are described in “Preparing configuration XML files” on page 4-4.

### Note:

- a. For BigSQL you can only use `fq-import-remote-conf.xml`.
  - b. For Hadoop formats, you can only use `fq-import-remote-conf.xml` for import and `fq-export-remote-conf.xml` for export.
2. Export the classpath for the Hadoop job:
    - When running standard (local) import or export, the path depends on the service provider:

### For BigInsights 3:

When using Hive, run the following command:

```
export HADOOP_CLASSPATH="/opt/ibm/biginsights/hive/lib/*:/opt/ibm/biginsights/hive/conf:/fluidqueryLocal/nzjdbc3.jar"
```

When using BigSQL, run the following command:

```
export HADOOP_CLASSPATH="/home/bigsql/sql/lib/java/*:/opt/ibm/biginsights/hive/conf:/fluidqueryLocal/nzjdbc3.jar"
```

### For BigInsights 4:

When using Hive, run the following command:

```
export HADOOP_CLASSPATH="/usr/iop/current/hive-client/lib/*:/usr/ibmpacks/bigsql/4.0/hive/conf:/fluidqueryLocal/nzjdbc3.jar"
```

When using BigSQL, run the following command:

```
export HADOOP_CLASSPATH="/home/bigsql/sql/lib/java/*:/usr/ibmpacks/bigsql/4.0/hive/conf:/fluidqueryLocal/nzjdbc3.jar"
```

### For Hortonworks:

For versions earlier than 2.3.2, run the following command:

```
export HADOOP_CLASSPATH="/usr/hdp/current/hive-client/lib/*:/usr/hdp/current/hive-client/conf:/fluidqueryLocal/nzjdbc3.jar"
```

On Hortonworks 2.3.2 and above, run the following command:

```
export HADOOP_CLASSPATH="/usr/hdp/current/atlas-server/hood/hive/*:  
/usr/hdp/current/hive-client/lib/*:/usr/hdp/current/hive-client/conf:/fluidqueryLocal/nzjdbc3.jar"
```

#### For Cloudera 4:

Run the following command:

```
export HADOOP_CLASSPATH="/etc/hive/conf:/usr/lib/hive/lib/*:  
/usr/share/cmf/cloudera-navigator-server/libs/cdh4/*:/fluidqueryLocal/nzjdbc3.jar"
```

#### For Cloudera 5:

Run the following command:

```
export HADOOP_CLASSPATH="/etc/hive/conf:/opt/cloudera/parcels/CDH/lib/hive/lib/*:/fluidqueryLocal/nzjdbc3.jar"
```

#### For Cloudera QuickStart:

Run the following command:

```
export HADOOP_CLASSPATH="/etc/hive/conf:/usr/lib/hive/lib/*:/fluidqueryLocal/*"
```

- When running remote import or export the command is the same for all providers:

```
export HADOOP_CLASSPATH=/fluidqueryLocal/nzjdbc3.jar
```

3. Optionally, you can validate connection configuration as described in “Validating data movement configuration” on page 4-28.
4. From the command line, run `nzcodec.jar` and provide the import or export XML configuration file as the `-conf` parameter. For example: To import data, run the following command:

```
hadoop jar /fluidqueryLocal/nzcodec.jar -conf fq-import-conf.xml
```

To export data, run the following command:

```
hadoop jar /fluidqueryLocal/nzcodec.jar -conf fq-export-conf.xml
```

**Tip:** You can also overwrite any parameters included in the configuration file with `-D` parameter:

```
hadoop jar /fluidqueryLocal/nzcodec.jar -conf fq-import-conf.xml -D nz.fq.tables=import_test -D  
nz.fq.append.mode=overwrite
```

## Results

Data transfer is performed based on the properties that you provided.

## Running data movement from NPS systems

Starting with version 1.5, IBM Fluid Query offers the capability to run data movement from NPS systems.

## Before you begin

**Note:** If your NPS system uses Kerberos authentication for its database user accounts, note that you cannot run the data movement feature from the NPS host to import or export files. As an alternative, you can run the data movement feature from the Hadoop service provider.

Install the IBM Fluid Query data connector software bundle on each NPS host. It provides important commands for setting up data movement. See “Installing IBM Fluid Query on NPS” on page 2-4.

## Procedure

1. Download the required Hadoop libraries, Hive libraries or BigSQL libraries, and client configuration files to the NPS system. On the NPS system, you can use the **fqDownload.sh** script to download the files. The command requires that the Hadoop software is installed in its default vendor locations on the service provider systems.

**Note:** Cloudera 5.4 and above, Hortonworks 2.3 and above, BigInsights 4.1 and above use libraries compiled with Java 7. As a result, running data movement from NPS against these systems requires IBM Netezza Analytics 3.2.1 (with Java 7 embedded) installed on NPS. Alternatively, you can use libraries obtained from older Hadoop versions (compiled with Java 6).

2. Using any text editor, edit the `fluidquery/conf/fq-remote-conf.xml` file that has the sample configuration. Save it as a new file name on the NPS filesystem and fill in the fields. The XML configuration details are described in “Preparing configuration XML files” on page 4-4.
3. Create a connection configuration using the **fqConfigure.sh** script. The command supports the data movement service type `fqdm` and the `--fqdm-conf /path/to/previously/edited/fq-remote-conf.xml` option to specify the configuration file that you created in Step 2. A sample command follows:

```
./fqConfigure.sh --service fqdm --provider ibm --fqdm-conf /<your_path>/fq-remote-conf.xml  
--config fqdm
```

4. Register two functions that you can use for importing files to and exporting files from the Hadoop service provider. Use the **fqRegister.sh** script to define and register the functions in one call. Separate the two function names with a comma using the form `toHadoop,fromHadoop` in the command.

```
./fqRegister.sh --config fqdm --udtf toHadoop,fromHadoop
```

**Note:** `toHadoop` and `fromHadoop` are default predefined names for the import and export function, and these names are used throughout this documentation. However, you can register these functions with different names.

5. Users who have execute access to the functions can run import and export by calling the functions from their applications or queries.
  - To import a table from NPS to Hadoop:

```
call toHadoop('nps_db', 'nps_table', 'hive_schema', 'hive_table');
```

Note that the first four arguments have a fixed order, although you do not have to provide the value if you want to use default values:

```
call toHadoop('nps_db_name', 'nps_table_name');
```

However, if you want to skip any parameters within the first four places, you must leave a placeholder:

```
call toHadoop('', 'nps_table', '', 'hive_table');
```

You can also override the XML file parameters using command arguments starting at the fifth argument:

```
call toHadoop('nps_db', 'nps_table', 'hive_schema',  
'hive_table', 'nz.fq.splits=12', 'nz.fq.fs.temp=/tmp');
```

If only some changes are needed in the XML configuration file, but the default NPS database, Hive schema, and Hive table should be used, you can omit those arguments from the command, for example:

```
call toHadoop('', 'nps_table', '', '', 'nz.fq.splits=12', 'nz.fq.fs.temp=/tmp',  
'nz.fq..', 'nz.fq...', ...)
```

- To export a table from Hadoop to NPS:

```
call fromHadoop('nps_db', 'nps_table', 'hdfs/input/path');
```

Note that in the export function, the first three arguments are fixed. You do not have to specify them if you want to use default values, but as for the `toHadoop` function, you need to leave placeholders if you want to skip any value within these three arguments:

```
call fromHadoop('nps_db', '', 'hdfs/input/path');
```

You can also override the XML file parameters using command arguments starting at the fourth argument:

```
call fromHadoop('nps_db', '', 'hdfs/input/path', 'nz.fq.splits=12');
```

For more information on these functions, see “The **toHadoop** and **fromHadoop** functions” on page 6-14.

## Running data movement from other systems

Starting with version 1.5, IBM Fluid Query offers the capability to run data movement from any system that runs Java.

### Before you begin

Depending on what kind of transfer you want to run, configure one of the XML configuration files.

### Procedure

1. From the Hadoop cluster, download the client configuration files, such as `core-site.xml`, `yarn-site.xml`, `hdfs-site.xml`, `mapreduce-site.xml`.
2. Download the Hadoop client libraries and Hive client libraries.

3. Download `nzjdbc3.jar` and `nzcodec.jar`.
4. In the XML configuration file, set the value for the **`nz.fq.hadoop.client.configs`** parameter. The value specifies the path to the downloaded client configuration files:

```
<property>
  <name>nz.fq.hadoop.client.configs</name>
  <value>/path/to/client/config/files/</value>
</property>
```

5. In the XML configuration file, set the value for the **`nz.fq.hadoop.remote.user`** parameter. The value specifies the remote Hadoop user name that will run data movement. This user must have access to HDFS.

```
<property>
  <name>nz.fq.hadoop.remote.user</name>
  <value>biadmin</value>
</property>
```

6. You can now launch data movement from any system that runs JVM. Type the following command (as one line) on that system:

```
java -cp '/path/to/hadoop/and/hive/client/libs/*:./nzcodec.jar:./nzjdbc3.jar'
com.ibm.nz.fq.NzTransfer -conf /path/to/fq-import.xml'
```

**Note:** `fq-import.xml` is the XML configuration file that is used in this example. It can be as well the `fq-export.xml` file for the export process.

---

## Importing or exporting multiple tables at a time

With IBM Fluid Query data movement feature, you can import or export a number of tables at a time.

### About this task

You can use the **`nz.fq.tables`** property in both, import and export configuration XML files to define a list of tables to be moved. As a value for this parameter, you provide a comma-separated list of NPS tables, that is, source tables for import, and target tables for export.

```
<property>
  <name>nz.fq.tables</name>
  <value>ADMIN.tab</value>
</property>
```

The format of this value is **`<SCHEMA>.<table>`**.

You can also use an asterisk (\*) character in the **`nz.fq.tables`** property for a combination of multiple schemas and tables, for example:

- `S1.T1` - imports table T1 in schema S1
- `S1.*` - imports all tables from schema S1
- `*,T1` - imports all tables T1 from all schemas
- `*,*` - imports all schemas with all tables
- `*` - imports all schemas with all tables

**Note:**

1. Include double quotations around delimited table names.
2. If full schema support is enabled on your NPS system, make sure to provide the schema name together with the table name in this property.
3. You cannot use \* as part of the schema or table name and use it as a filter, for example: S1.a\*, meaning "all tables in schema S1, with name starting with a". The system cannot interpret such filter, and a\* is interpreted as the exact table name.

## Procedure

- To import multiple tables from NPS to Hadoop:
  1. Define the directory where the data should be stored on Hadoop with parameter **nz.fq.output.path**.
  2. List source table names from NPS using **nz.fq.tables** parameter. You can use asterisk to filter the tables.
  3. Proceed with configuration and run import as usual.
- To export multiple tables from Hadoop to NPS:
  1. Define the source tables directory on Hadoop in **nz.fq.input.path**.
  2. List source table names in **nz.fq.tables** parameter. You can use asterisk to filter the tables.
  3. Proceed with configuration and run export as usual.

---

## Importing data to Hadoop with views created on NPS

Data federation is a process where data is collected from distinct databases without copying or transferring the original data itself. This allows you to aggregate data from disparate sources, such as NPS and Hadoop.

Starting with version 1.7.1, you can automatically create views on NPS of the data that you import to Hadoop. The functionality is called automatic federation and you can activate it by adding additional parameter to the toHadoop import function.

## Before you begin

Automatic federation can only be configured for import executed directly from NPS, that is, by using the toHadoop function.

The procedure is almost the same as the one described in "Running data movement from NPS systems" on page 4-32, but you need to specify an additional parameter **--auto-connector** when running fqConfigure.sh.

## Procedure

1. When configuring the toHadoop function on NPS, add the following parameter with the value of your choice: **--auto-connector <user\_defined\_suffix>**. For example:

```
./fqConfigure.sh --service fqdm --provider ibm --fqdm-conf conf/fq-remote-conf-import.xml
--auto-connector federated_view
```

2. Register the function for importing files with the fqRegister.sh script. By default, the function is registered with the toHadoop name, but you can change it using the **--udtf <function\_name>** parameter.
3. Run the registered import function from NPS:

```
call toHadoop('db_name','table_name');
```

## Results

When import completes, the following message is displayed:

Fluid Query Data Movement finished successfully.  
Auto connector enabled. View(s) with suffix *toHadoop\_federated\_view* were created.

For each imported table, a view is created on NPS. The name of the view consists of the following parts:

<imported\_table\_name>\_<function\_name>\_<user\_defined\_suffix>

**Note:** If the view with the same name already exists on NPS, and the append mode in the configuration XML file is set to **nz.fq.append.mode=create**, the import fails with an error message.

## What to do next

For each import operation that you run with the registered function, a view will be created on NPS automatically.

If you want to run the import operation with the registered function but without creating the view, you must add an empty parameter `fq.fqdm.auto.connector` as the fifth argument to the function, for example:

```
call toHadoop('db_name','table_name','', '','fq.fqdm.auto.connector=');
```

---

## Importing data to partitioned and clustered Hive tables

Starting with IBM Fluid Query 1.7.1, you can import tables to Hadoop taking advantage of the Hive partitioning and clustering features.

### Partitioning

Using Hive, you can organize tables into partitions. It is a way of dividing a table into related parts based on the values of partitioned columns such as date, city, and department. Using partition, it is easy to query a portion of the data.

### Clustering

You can also sub-divide tables into the so-called *buckets*, to provide extra structure to the data that may be used for more efficient querying. Clustering works based on the value of hash function of some column or columns of a table.

Clustered tables decrease time of execution of queries with join clause in Hive tables. You provide one or more columns, and a number of buckets for clustering. Buckets are files that contain data, they are created in HDFS.

### Limitations

The following restrictions and limitations apply to using Hive partitioning and clustering with IBM Fluid Query:

- Partitioning is dependent on your Hadoop cluster resources. With big volume of data and high number of partitions the import might fail on insert into destination table (the 2nd stage of import algorithm, as described in “Data movement algorithm” on page 4-2). To troubleshoot such problems you can import data to non-partitioned table using IBM Fluid Query and then manually INSERT SELECT from this table to the destination partitioned table. This approach helps to resolve Hadoop cluster configuration or resource issues.
- You can only use remote mode of import and export.
- The use of BigSQL is not supported. You must specify Hive2 as a service for fq.sql.type property in the configuration file.
- Partitioning and clustering is not available for data in NZBAK (Netezza compressed) format.
- While querying works faster, the import operation to clustered or partitioned tables might take longer.
- You can only import one table at a time.
- When using partitioning, you cannot specify all columns that exist in a table that you want to import. This is a limitation on Hive side. When you specify all columns in a table for partitioning, the error message is displayed.

## Usage

To import a table from NPS to Hadoop you use the configuration XML file for remote mode. In the file template, there are new properties available:

- For partitioning:

```
<property>
  <name>fq.hive.partitioned.by</name>
  <value></value>
  <description>Column(s) in a table that will be used for partitioning</description>
</property>
```

The fq.hive.partitioned.by property determines by which column(s) the imported table will be partitioned. You can provide multiple columns divided by comma. By default the field is empty, that is, partitioning is disabled.

**Note:** You cannot specify all columns that exist in a table for partitioning. You must exclude at least one column. This is a limitation imposed by Hive.

- For clustering:

```
<property>
  <name>fq.hive.clustered.by</name>
  <value>id</value>
  <description>Column(s) in a table that will be used for clustering. To enable default
clustering, keep this property empty and set fq.hive.cluster.buckets.</description>
</property>
<property>
  <name>fq.hive.clustered.buckets</name>
  <value>252</value>
  <description>Integer value which determines the number of buckets that will be created
during clustering. This property cannot be empty when using clustering.</description>
</property>
```

There are two ways in which you can set clustering:

1. You can use default clustering that is, use the same value that was set for the table on NPS with **DISTRIBUTE ON** parameter. To use default clustering, leave **fq.hive.clustered.by** empty and only set a number of buckets in **fq.hive.clustered.buckets**.



2. You can cluster by specific columns of your choice. To create such explicit distribution key, provide one or more column names in **fq.hive.clustering.by**. Also, set the number of buckets in **fq.hive.clustering.buckets**.

Exporting a clustered or partitioned table to NPS does not require any additional settings.

---

## Exporting data from Hadoop

Depending on what type of data you plan to transfer from Hadoop, and whether the data on Hadoop was imported from NPS, you can follow one of the three export scenarios.

### About this task

The scenarios are described in detail in the following sections.

1. You can export data that was previously imported from NPS, as described in “Exporting previously imported files”
2. You can export text files that were not previously imported from NPS, as described in “Exporting text files from Hadoop” on page 4-40. This option is available starting with version 1.7.
3. You can export a Hive table (in any format) that was not imported from NPS, as described in “Exporting a Hive table that was not imported from NPS” on page 4-41. This option is available starting with version 1.7.1.

All of the mentioned export types can be run from both, NPS - using the **toHadoop** and **fromHadoop** functions, and from Hadoop - using the `fdm.sh` script.

**Note:** When exporting new text data from Hive, you can either choose to export text files (scenario 2), or to export a table (scenario 3). Exporting a text file is better from performance point of view, however it requires more information about the data (table structure, separator, null representation etc.), and an additional manual step of creating the table on NPS.

## Exporting previously imported files

Follow these guidelines for exporting files that were previously imported from NPS.

### About this task

Note that any restrictions that were applicable for import (for example, using remote import mode) are also applicable when exporting the same files.

If you want to export files that were previously imported using IBM Fluid Query, you can do it in two ways:

- Use the HDFS directory of a table as **nz.fq.input.path**, for example:

`<import_set_path>/<table_name>`

In this case, IBM Fluid Query will use the metadata created during import. This metadata contains the table definition and allows recreating the table if it does not exist.

- Use the HDFS directory where data files are located, for example:

```
<import_set_path>/<table_name>/table
```

In this case, the table must exist in NPS.

**Note:**

- This procedure does not work for Hadoop formats. If you are exporting data in Hadoop formats see Exporting data stored in Hadoop formats.

## Exporting text files from Hadoop

Starting with IBM Fluid Query version 1.7, you can export text files from Hadoop to NPS, even if those files were not previously imported from NPS.

### About this task

If you have text files on Hadoop that fit the existing table structure on NPS, you can transfer these files to NPS. If such structure does not exist, you can create the target table on NPS that will match the structure of text data on Hadoop, and then transfer the files.

You can either run the export operation from Hadoop, using the `fdm.sh` script, or from NPS, using the **fromHadoop** function. This topic provides detailed instruction using the first method.

### Procedure

1. Verify the structure of your text data on Hadoop:
  - If the table exists in Hive, run the **describe formatted** command. The output contains details on your Hive table definition.
  - If the table does not exist in Hive, you must check HDFS location and the structure of the text files, such as fields, separators, null representation.
2. On NPS, locate, or manually create a table to which you will export the data.

**Note:**

- a. Table structure must be the same as the data structure in the text file on Hadoop.
  - b. You must verify the value for field delimiter (**nz.fq.format.fielddelim**), and the representation of NULL for text columns (**nz.fq.format.null**). You can find this information in Hive table definition, or in text files structure.
3. On Hadoop, edit the available template for export configuration XML, and specify at least the following values:
    - **nz.fq.input.path** - This parameter must point to the path where the text files are located on Hadoop;
    - **nz.fq.table** - This parameter must point to the target table on NPS.
    - **nz.fq.format.fielddelim** - Look for `field.delim` information in the Hive table definition. In the configuration XML you must specify an ASCII code for the character. If there is no specification for `field.delim` on Hive, use default value, that is 1.
    - **nz.fq.format.null** - Look for `serialization.null.format` in Hive table definition and provide the same value in the XML file. If it is not specified in the output, set the value to `\N`, which is default for Hadoop.
  4. Specify any other required parameters and save the XML file.

5. On Hadoop, run the command:

```
fdm.sh -conf <configuration_file>.xml
```

**Tip:** If required, you can overwrite any parameters included in the configuration file with the **-D** parameter, for example:

```
fdm.sh -conf fq-export-conf.xml -D nz.fq.table=test -D nz.fq.format.fielddelim=124
```

## Results

Text files are transferred from Hadoop to the specified NPS table.

## Exporting a Hive table that was not imported from NPS

Starting with IBM Fluid Query 1.7.1, you can export a Hive table from Hadoop to NPS, even if the table was not previously imported from NPS. You can also use the SQL WHERE clause when exporting to narrow down the results.

### Before you begin

This type of export does not require any metadata information. The table schema is created based on Hive table description. All existing export modes (create, overwrite, append) are supported, however for create and append modes, if the table already exists on NPS, it must have the same schema as the schema created for Hive table.

The following table presents data type conversion from Hive to NPS:

*Table 4-7. Data type conversion*

Hive type	NPS type
INT	INTEGER
SMALLINT	SMALLINT
TINYINT	BYTEINT
BIGINT	BIGINT
DECIMAL (p,s) <b>Note:</b> See <b>Limitations</b> following the table.	NUMERIC (p,s)
STRING	NATIONAL CHARACTER VARYING (max(length())) <b>Note:</b> String data type is converted to NVARCHAR on NPS. The length is calculated as the maximum length of all values inside the column.
CHAR (x)	NATIONAL CHARACTER (x)
VARCHAR (x)	NATIONAL CHARACTER VARYING (x)
BOOLEAN	BOOLEAN
DOUBLE	DOUBLE PRECISION
FLOAT	REAL
DATE	DATE
TIMESTAMP	TIMESTAMP

## Limitations

- DECIMAL(Precision, Scale) data type is only supported for Hive versions 0.13 and above. Versions prior to 0.13 (for example, on BigInsights 3) use DECIMAL with other structure, and export of a table with column type DECIMAL fails with error.
- This method of export only works in remote mode, so JDBC access to Hive is required. Also, you must use fq-export-remote-conf.xml template for configuration, and specify the JDBC properties there.
- NPS only supports UTF-8 coding.
- The following characters are not supported in column name of table which is to be exported:
  - : (colon)
  - . (full stop)
  - ` (acute accent)
  - , (comma)

Behavior may vary depending on your Hadoop distribution or Hive version, but most likely export of such table will fail.

- If the Hive table that you want to export contains text data, from performance point of view it might be more efficient to use the method described in “Exporting text files from Hadoop” on page 4-40.

## About this task

You can run the export operation either from NPS using the fromHadoop function, as in the following example, or from Hadoop using the fdm.sh script. Remember to use the remote mode, and fq-export-remote-conf.xml template.

## Procedure

1. Create a copy of the fq-export-remote-conf.xml template and edit it as required.
2. Run the function that is registered for exporting data from Hadoop to NPS, for example:

```
call fromHadoop('nps_db','tab1','', 'fq.input.path=', 'fq.hive.tablename=tab1');
```

You must specify at least the following parameters, where the first three have a fixed order:

- 1st parameter 'nps\_db' is a destination database on NPS,
- 2nd parameter 'tab1' is the destination table on NPS,
- 3rd parameter '' stands for HDFS path and it must be empty,
- 'fq.input.path=' must be listed in the command and the value must be empty,
- 'fq.hive.tablename=tab1' is the name of the Hive table that you want to export.

Optionally, you can also set the following parameters related to Hive table export:

- If you want to filter the data on Hive table with the SQL WHERE clause before exporting it, you can use the fq.hive.where property.

- If you want to specify an explicit distribution key for the exported table, provide one or more column names as values for `fq.hive.set.distribute.on`. By default, the value is empty and distribution is random.
- You can set `fq.skip.table.structure.validation=true` if you want to disable the schema validation between source and destination tables. By default, schema on NPS is validated and it must contain NVARCHAR, not VARCHAR. Moreover, string length on Hive is calculated during export, as all string fields in Hive table are mapped to NVARCHAR with size set to `max(length(col))`. However, if you know that the strings in your Hive data can be converted to VARCHAR, you can set `fq.skip.table.structure.validation=true`, which means table structure validation and string column length calculation will be omitted.

---

## Querying compressed NZBAK files from Hadoop

After you import data from NPS and store it on Hadoop in NZBAK format, perform additional configuration steps to be able to query the data.

### About this task

Before you can query NZBAK compressed files, you must configure your Hadoop system so that it can detect the `nzcodec.jar` file.

**Note:** In configurations where Hadoop services are located on more than one node, you must install the data movement feature on each node that has services that could access the compressed binary files. The `nzcodec.jar` file and other required data movement files must be stored in the local filesystem in order to read the compressed binary files.

## Querying compressed NZBAK files on BigInsights 3

To enable querying compressed files on IBM BigInsights 3 with Hive, you must first perform a set of configuration steps on your Hadoop environment.

### Procedure

1. Edit the file `hive-env.sh`. The file is usually located in `/opt/ibm/biginsights/hdm/components/hive/conf/`
2. In the `hive-env.sh` file, add the following line to **HIVE\_AUX\_JARS\_PATH**:

```
HIVE_AUX_JARS_PATH: /fluidqueryLocal/nzcodec.jar
```

**Important:** Paths must point to the directory on the local file system where the `nzcodec.jar` file is located.

3. Deploy the configuration. Run the following command:

```
/opt/ibm/biginsights/bin/synconf.sh hive
```

4. Restart Hive and BigSQL.

## Querying compressed NZBAK files on BigInsights 4

To enable querying compressed files on IBM BigInsights 4, you must first perform a set of configuration steps on your Hadoop environment.

## About this task

Perform these steps from the web user interface:

### Procedure

1. Go to **Hive > Configs > Advanced hive-env**.
2. At the beginning of the **hive-env** template, add the following property:

```
HIVE_AUX_JARS_PATH="/fluidqueryLocal/nzcodec.jar"
```

**Important:** Paths must point to the directory on the local file system where the `nzcodec.jar` file is located.

3. Save all changes and restart Hadoop.

## Querying compressed NZBAK files on Hortonworks

To enable querying compressed files on Hortonworks with Hive, you must first perform a set of configuration steps on your Hadoop environment.

### About this task

Perform these steps from the web user interface.

### Procedure

1. Go to **Hive > Configs > Advanced hive-env**.
2. At the beginning of the template, add the following property, specifying a particular file name:

```
HIVE_AUX_JARS_PATH="/fluidqueryLocal/nzcodec.jar"
```

3. Go to **Hive > Configs > Custom hive-site** and add **hive.aux.jars.path > /fluidqueryLocal/nzcodec.jar**.
4. Save all changes and restart Hadoop.

## Querying compressed NZBAK files on Cloudera

To enable querying compressed files on Cloudera, you must first perform a set of configuration steps on your Hadoop environment.

### About this task

**Restriction:** Querying files in NZBAK compressed format is not supported for Impala. For this task you can use Hive interface only.

### Procedure

1. In Cloudera Manager, edit the configuration for the Hive service. Go to **AUX. > Hive Auxiliary JARs Directory** and modify the value to **/fluidqueryLocal**.

**Note:** The path must point to the directory on the local file system where the `nzcodec.jar` file is located.

2. Add the same path to the **hive.aux.jars.path** property in the HiveServer2 Configuration Safety Valve for `hive-site.xml`. This allows the JARs to be passed to MapReduce jobs spawned by Hive:

```
<property>
  <name>hive.aux.jars.path</name>
  <value>file:///fluidqueryLocal/nzcodec.jar</value>
</property>
```

3. Save the changes and deploy the new configuration.
4. Restart the cluster.

## Results

The configuration steps are complete.

If you are using a Cloudera 4.7 configuration, steps 1 and 2 above are different. For Cloudera 4.7 installations, go to **Hive configuration >service-wide > Advanced** and set the following options:

- In **Hive Service Configuration Safety Valve for hive-site.xml**, add the value:

```
<property>
  <name>hive.aux.jars.path</name>
  <value>file:///fluidqueryLocal/nzcodec.jar</value>
</property>
```

- In **Hive Service Environment Safety Valve**, add the value:

HIVE\_AUX\_JARS\_PATH=/fluidqueryLocal/nzcodec.jar

Save your changes and deploy the configuration, and restart the cluster.

## Querying compressed NZBAK files on BigSQL

If you are using BigSQLv1 or BigSQLv3, querying of the compressed files is configured automatically during the installation of IBM Fluid Query.

### About this task

After the installation, make sure to restart BigSQL so that querying of compressed files is properly configured. If the configuration has changed after the installation of IBM Fluid Query, for example a new node was added, you may need to perform the following procedure to enable querying compressed data.

### Procedure

1. Log in as the bigsql user to the master node.
2. Re-run the following script:

```
/fluidqueryLocal/scripts/inst_bigsql.sh
```

In case IBM Fluid Query was installed to a non-default directory, run the following command instead:

```
<PATH_TO_IFQ>/scripts/inst_bigsql.sh --jar <PATH_TO_IFQ>
```

3. As the bigsql user, restart BigSQL:
  - a. Run `$BIGSQL_HOME/bin/bigsql stop`
  - b. Run `$BIGSQL_HOME/bin/bigsql start`

## Data type mapping for data movement

When the tables are transferred from NPS to Hadoop, some of the data types are changed. The following table shows the data conversion pattern.

Table 4-8. Data type mapping for data movement

Data type on NPS	Data type on Hive	Data type on BigSQL
Boolean	Boolean	Boolean
Real, Double	DOUBLE, FLOAT	DOUBLE, FLOAT
Byteint, Smallint, Integer, Bigint	TINYINT, SMALLINT, INT, BIGINT, BIGINT	TINYINT, SMALLINT, INT, BIGINT, BIGINT
Numeric	DECIMAL for Hive 0.13 and above; STRING for older Hive versions	DECIMAL
Char, Varchar, Nchar, Nvarchar	STRING - default, see note below;  VARCHAR when: <b>fq.hive.usevarchar = true</b> in the import configuration XML file, and Hive version is 0.12 and above.	VARCHAR
Date	STRING - default  DATE when <b>nz.fq.hive.usedate = true</b> in the import configuration XML, and Hive version is 0.12 and above.	VARCHAR(30)
Timestamp	TIMESTAMP	TIMESTAMP
Time	STRING	VARCHAR(30)
Interval	STRING	STRING

**Note:**

1. For data types that are mapped to STRING, such as date, time, and interval, the NPS null value is stored as an empty string on Hadoop.
2. Because of the limitations described in “Limitations on VARCHAR data types” on page 5-11, on Hive, the VARCHAR data types are by default changed to STRING.



---

## Chapter 5. Troubleshooting

Read about known issues, common errors and learn how to troubleshoot problems.

---

### Resolving problems after downgrading IBM Fluid Query to version 1.5

It is important to know that when you downgrade IBM Fluid Query to any version prior to 1.6, the Java™ version is also changed by the installer, and it is incompatible with IBM Netezza Analytics 3.2.1. Even if you then upgrade to 1.6 again, the Java version remains incompatible. You must perform recovery steps to revert the changes.

**Note:** Downgrading IBM Fluid Query to any version prior to 1.6 is not supported when using IBM Netezza Analytics 3.2.1 and higher.

#### About this task

IBM Fluid Query 1.5 installer creates a symbolic link to Java shipped within IBM Fluid Query and it backs up the existing IBM Netezza Analytics Java to EXISTING\_JAVA\_NAME.orig. To recover the original version of Java that was shipped with IBM Netezza Analytics, perform the following steps:

#### Procedure

1. Change directory:

```
cd /nz/export/ae/languages/java/java_sdk
```

2. Run the following command:

```
ls -l
```

Verify that the following directories exist:

- ibm-java-i386-71.orig

Note that the name of this directory might differ depending on the version of IBM Netezza Analytics that is installed.

- A symbolic directory with the same name but without the .orig suffix.

3. Remove the symbolic link. Note that the directory name might be different.

```
rm ibm-java-i386-71
```

4. Change the name of the remaining Java directory by removing the .orig suffix:

```
mv ibm-java-i386-71.orig ibm-java-i386-71
```

#### Results

IBM Netezza Analytics now uses its original Java version.

---

### Data connector troubleshooting

Find out how to troubleshoot problems when using data connector.

## Data connector limitations

Read about limitations when using IBM Fluid Query data connectors.

- It was observed, that generic connector to Sybase IQ does not work as expected on big data when using driver jConn.
- For dashDB and Oracle, most data types are supported, although some read operations might not work if the size of a column is too large for NPS to handle. For more information about data types, see Data type mapping for predefined connectors.

## Data connector known issues

Note the following important behaviors and limitations for the data connector feature:

- Do not attempt to remove the IBM Fluid Query feature by deleting or removing the `/nz/export/ae/products/fluidquery` directory or any of its contents. Deleting these files might affect the operation of other features such as IBM Netezza Analytics.
- There is no `--help` option available for the **fqRemote.sh** script.
- If you specify an invalid user in the **fqRegister.sh** script, the system displays an extraneous warning for the `--path` option, for example:  
WARNING: --path should start with /nz/export/ae/applications/test\_db  
nzsql: Password authentication failed for user user1  
ERROR running SQL:  
Registering functions and credentials in database "test\_db" failed.

The error message for the password authentication is valid, but you can ignore the `--path` message.

- When using the SELECT command to read from Impala tables in a non-default database, the select command can sometimes fail with the following error:  
TEST.ADMIN(ADMIN)=> SELECT \* FROM TABLE WITH FINAL ( my\_cloudimpala1('my\_test', '', 'select \* from tbl\_1')) ;  
ERROR: Unable to determinate shape  
Instead of specifying the database value, you can use a *db\_name.table\_name* format and the SQL command will work, for example:  
TEST.ADMIN(ADMIN)=> SELECT \* FROM TABLE WITH FINAL ( my\_cloudimpala1('', '', 'select \* from my\_test.tbl\_1')) ;
- If you create a view as described in “Using views to simplify user queries” on page 3-31, note that there is a known issue for views that reference user-defined table functions such as the FqRead() functions. The views work for user queries, but the view definition has incomplete information. The problem prevents the **nzrestore** command and restore operations to fail to restore the view in NPS releases 7.0.4.x, 7.1.x, and 7.2.x. If you restore a database that includes these UDTF-based views, you must manually recreate each view that calls the data connector functions, and grant the user privileges to each view. The problem is resolved in NPS release 7.0.4.7-P1 and later, 7.1.0.4-P1 and later, and 7.2.0.3-P1 and later.
- For dashDB and Oracle, most data types are supported, although some read operations might not work if the size of a column is too large for NPS to handle. For more information about data types, see “Data type mapping for predefined connectors” on page 3-7.
- If you run multiple parallel queries on Cloudera, you might get the following error:

```
java.sql.SQLException: [Cloudera][JDBC](10140) Error converting value to long.
```

If you were using Cloudera official Hive JDBC drivers available on the web, try installing the drivers from Cloudera Distributed Hadoop instead.

- If you use more than one Hadoop authenticated by Kerberos on remote mode, you must use IBM Netezza Analytics with Java 1.7, that is version 3.2.1 or higher. Otherwise, you might get the following error when running a query:

```
ERROR: Unable to authenticate with Kerberos
```

- If you query a table on DB2 that contains invalid data, you might get errors such as:

```
ERROR: [jcc][t4][1065][12306][3.63.123] Caught java.io.CharConversionException.
```

```
ERROR: [jcc][t4][1065][12306][3.63.123] Caught java.io.CharConversionException. See attached Throwable for details. ERRORCODE=-4220, SQLSTATE=null
```

To solve the issue, you can set the JCC configuration property `db2.jcc.charsetDecoderEncoder=3` so that instead of throwing an exception the JCC driver returns the Unicode REPLACEMENT CHARACTER (U+FFFD) in place of a sequence of bytes that is not a valid UTF-8 string. To do this, add the `--addJavaParam` with value `-Ddb2.jcc.charsetDecoderEncoder=3` to the `fqRegister.sh` command, as in the following examples:

```
./fqRegister.sh --addJavaParams -Ddb2.jcc.charsetDecoderEncoder=3 --debug
--config IFQ_BI --udtf toQuote,fromQuote --db \"TEST_quotes\"

./fqRegister.sh --addJavaParams -Ddb2.jcc.charsetDecoderEncoder=3
-Djava.test=test -Djava.test2=test2 --debug --config IFQ_BI --udtf toQuote,fromQuote
--db \"TEST_quotes\"
```

---

## Data movement troubleshooting

Refer to the following section if you are experiencing problems with the data movement feature.

### Log files

When data export or import fails, you might refer to the following log files for mappers:

- on Hadoop: `/fluidqueryLocal/var/log/mapper`
- on Netezza: `/nz/export/ae/products/fluidquery/logs/mapper`

### Error: `java.lang.ClassNotFoundException: org.netezza.Driver`

**Solution:** Before starting IBM Fluid Query, you must add the path to the `nzjdbc3.jar` file to the `HADOOP_CLASSPATH` variable. To connect to Hive directly, you must add the path to the Hive `lib` directory and to its configuration. For a description of how to add the path for most of the Hadoop distributions, see “Deprecated: Running data movement from Hadoop with `nzcodec.jar`” on page 4-30.

**Errors: Exception: hive-site.xml cannot be found in classpath and java.lang.NoClassDefFoundError: org/apache/hadoop/hive/q1/...**

**Solution:** Add directories that contain hive-site.xml and Hive jars to HADOOP\_CLASSPATH. For example:

```
export HADOOP_CLASSPATH="/etc/hive/conf/:/usr/share/cmf/cloudera-navigator-server/libs/cdh4/*"
```

**Error: java.lang.ClassNotFoundException: com.ibm.nz.codec.NzInputFormat**

**Solution:** You must define the nzcodec.jar file as a custom input format class for each service that reads IBM PureData System for Analytics compressed binary format files.

The process for adding a custom input format file varies for each Hadoop service. Refer to “Querying compressed NZBAK files from Hadoop” on page 4-43 for examples of how to add the file for Hive. You might need to define the nzcodec.jar location to other Hadoop service providers that you use.

For example, if you use the Hive CLI, you might need to configure the location of the jar file using a command similar to the following:

```
add jar /fluidqueryLocal/nzcodec.jar
```

For the Hive server, you could add the file by changing the hive-site.xml file (either manually or by using an interface such as the Cloudera Manager) and adding the following section:

```
<property>
  <name>hive.aux.jars.path</name>
  <value>file:///fluidqueryLocal/nzcodec.jar</value>
</property>
```

Refer to the documentation for your Hadoop service for specific instructions to add custom input format files to the service classpath.

**Error: Unable to get transaction ID!ERROR: Permission denied on "\_VT\_DBOS\_CONNECTION" or "\_VT\_DISK\_PARTITION".**

**Solution:** The NPS user must have permissions to read from the database and table but to retrieve the information, the following privileges are also required:

```
GRANT select,list on _VT_DBOS_CONNECTION TO _user_
GRANT select,list on _VT_HOSTTX_INVISIBLE TO _user_
GRANT select,list on _V_SYS_OBJECT_DSLICE_INFO TO _user_
GRANT select,list on _VT_DISK_PARTITION TO _user_
```

The NPS user also needs rights to create external tables:

```
GRANT external table TO _user_
```

Access can be checked by running queries from the user session:

```
SELECT * FROM _VT_DBOS_CONNECTION;
SELECT * FROM _VT_HOSTTX_INVISIBLE;
```

## Error: Error during checking table compatibility ERROR: Permission denied on "\_T\_OBJECT".

**Solution:** When using the data movement feature, a Permission denied on "\_T\_OBJECT" error indicates that the user does not have Select privileges to the table that is required for validating object types during the file transfers. To correct this problem, grant the NPS database user Select permissions to read from the \_T\_OBJECT table.

```
GRANT SELECT ON _t_object To _user_
```

## Service does not start after configuring nzcodec.jar location

**Solution:** In configurations where Hadoop services are running on different nodes in a cluster, the data movement feature must be installed on each node that has a service that uses the nzcodec.jar to access NPS compressed binary files. The nzcodec.jar must be in local file system for the node. If the data movement feature is not installed on the Hadoop node and you follow the steps to add the nzcodec.jar to the service classpath, an error similar to the following could appear when you try to start the Hadoop service:

```
+ ERROR: HIVE_AUX_JARS_PATH is configured in Cloudera Manager as /fluidqueryLocal.
However directory /fluidqueryLocal does not exist. When configured, directory
specified in HIVE_AUX_JARS_PATH must be created and managed manually before
starting Hive. /usr/lib64/cmf/service/hive/hive.sh: line 107: ERROR:: command not
found + exit 1
```

Refer to the instructions in "Installing or upgrading the data movement feature" on page 2-8 for the instructions to install the data movement feature.

## Tables imported to BigInsights are not visible for BigSQLv3

When you set the following configuration XML property `nz.fq.sql.type=hive2`, the tables that were created in Hive might not be visible in BigSQLv3. To see the tables in BigSQLv3, you must call a sync objects procedure. Run the following commands:

1. **CALL SYSHADOOP.HCAT\_SYNC\_OBJECTS('schema\_name', '.\*', 'a', 'REPLACE', 'CONTINUE');**

For the **schema\_name** value, use the name of your database on Hive.

2. If you are running this procedure for a specific table, run:

```
CALL SYSHADOOP.HCAT_CACHE_SYNC('schema_name','table_name');
```

or, if you are running the procedure for all tables in the schema, run the following command:

```
CALL SYSHADOOP.HCAT_CACHE_SYNC('schema_name');
```

You can also set the configuration XML property as follows: (`nz.fq.sql.type=bigsql`) and use BigSQL in import, which will solve the problem permanently.

## Special characters support on Hadoop

### Special characters in table names

NPS supports UTF-8 special characters for the table names, while Hive only supports alphanumeric characters and underscores. When importing

tables with special characters in their names, make sure to use the **nz.fq.hive.tablename** property. This property defines the output table name in Hive.

### Special characters in column names

On Hive: Special characters in column names are supported in Hive version 0.13.0 and higher. In older versions of Hive, when importing a table with special characters in column names, set the **nz.fq.sql.metadata** property to **false**. The table will not be created in Hadoop but the table data will be imported. Open the `metadata.xml` file that holds this data and look for the **metadata.sql.create** property. The value of this property contains the table definition. Do not edit this definition in the `metadata.xml` file. Copy it, rename the columns, and run it in Hive to properly create the table. You can avoid this issue by upgrading Hive to version 0.13.0 or higher.

On BigSQL: Special characters in column names might also create problems for BigSQL. Following is a sample error when **!** is used in column name:

```
Caused by: com.ibm.nz.fq.common.exception.hive.HiveException: Unable to execute Hive sql.
DB2 SQL Error: SQLCODE=-7, SQLSTATE=42601, SQLERRMC=;!all_dt_boundval ( DI, DRIVER=3.70.4
at com.ibm.nz.fq.hive.RemoteHiveQueryRunner.executeUpdate(RemoteHiveQueryRunner.java:104)
at com.ibm.nz.fq.hive.DbEngine.executeUpdate(DbEngine.java:87)
at com.ibm.nz.fq.hive.DbManager.createTableInHive(DbManager.java:99)
at com.ibm.nz.fq.ArchiveManager.importDataToHadoop(ArchiveManager.java:86)
```

You might also get the following message:

```
Caused by: com.ibm.db2.jcc.am.SqlSyntaxErrorException:
DB2 SQL Error: SQLCODE=-104, SQLSTATE=42601, SQLERRMC=INT;st;, DRIVER=3.70.4
```

### JDBC driver compatibility

If you experience problems with compatibility of the JDBC driver, make sure to use the driver that is aligned with the version of NPS that you are using. To replace the JDBC driver, perform the following steps:

1. Obtain the version of the driver that is aligned with your NPS release.
2. On Hadoop, replace `nzjdbc3.jar` in the installation directory on the local system. By default, this directory is `/fluidqueryLocal`

### Defining separate users for Hive and HDFS

If for any reason you require separate users for HDFS and Hive operations, you can define the `nz.fq.hadoop.remote.user` parameter in the configuration XML file.

**Import fails on Hive with Error message: Cannot create table in Hive. Data has been imported properly. Try creating it manually.**

#### Solutions:

1. Ensure that the Hadoop user that is executing the import operation has permission for creating table in Hive.
2. If solution 1 does not work, verify in Hive service configuration that the **hive.server2.enable.doAs** parameter is set to true. If it is set to false, then change it to true.
3. If solution 2 cannot be implemented, then you must execute import operations with **nz.fq.sql.metadata** set to false and then, after import is completed, manually create table in Hive.

## Troubleshooting a full schema enabled system when target table name is not specified

On system with full schema you can encounter a problem with a specific schema and table names if the target (Hive) table name is not specified.

The following scenarios are possible:

- import is executed from NPS and the 4th argument of toHadoop function is not provided
- import is executed from Hadoop and `nz.fq.hive.tablename` is not specified
- import is executed from Hadoop and entire database is imported (`nz.fq.tables=*`)

In these cases Hive table name will be created automatically. This may lead to problems if in NPS database there are tables called as follows:

- `abc.def_ghi`, where `nps_schema` is `abc` and NPS table is `def_ghi`
- `abc_def.ghi`, where `nps_schema` is `abc_def` and NPS table is `ghi`

Both tables will have exactly the same target table name in Hive and the same path in HDFS if imported to Hadoop, which might lead to data loss.

### Fast data movement executed from NPS fails with error `java.lang.IllegalArgumentException: Unable to parse '/iop/apps/${iop.version}/mapreduce/mapreduce.tar.gz#mr-framework' as a URI, check the setting for mapreduce.application.framework.path`

This situation might happen on BigInsights. In order to solve the problem, replace `${iop.version}` with the actual version of BigInsights (to form the correct path on master node) in the XML files downloaded on NPS using `fqDownload.sh`.

## Limitations on decimal values in BigInsights

1. Import fails on BigInsights 3 if a table has a NUMERIC field. The field is mapped to DECIMAL, which is not supported in BigInsights 3. The following message is displayed:

Decimal type not supported. If you are using BigInsights 3 and you want to map DECIMAL to STRING, add "nz.fq.sql.bigsql.v3compatibility" property to the xml configuration file and set it to true.

To solve the problem, add the following property to the XML configuration file:

```
<property>
<name>nz.fq.sql.bigsql.v3compatibility</name>
<value>true</value>
</property>
```

2. BigInsights version 3 and version 4 cannot support imported NPS tables that contain larger-precision decimal values. Table creation in BigSQL fails with error similar to:

```
Exception in thread "main" com.ibm.nz.fq.common.exception.QArchException:
Cannot create table in Hive. Data has been imported properly. Try creating it manually
....
Caused by: com.ibm.nz.fq.common.exception.hive.HiveException: Unable to execute Hive sql.
DB2 SQL Error: SQLCODE=-604, SQLSTATE=42611, SQLERRMC=DECIMAL(38, 0), DRIVER=3.70.4
```

or

```
Failed to instantiate extended type class com.ibm.biginsights.catalog.type.DecimalType,
for type "decimal(38,0)"
```

The maximum precision for a decimal value in BigInsights version 3 or version 4 is 31 digits. The Netezza Platform Software and the Hive application both support a maximum of 38 digits of precision. For more information about the BigInsights data type support, see the IBM Knowledge Center.

A possible workaround is to import NPS tables that use decimal values that have a precision of 31 digits or less.

## Data movement known issues and limitations

Refer to this topic for information on the limitations of IBM Fluid Query data movement.

### Cannot run data movement from NPS system with Kerberos authentication

If your NPS system uses Kerberos authentication for its database user accounts, note that you cannot run the data movement feature from the NPS host to import or export files. As a alternative, you can run the data movement feature from the Hadoop service provider.

### GPFS data movement between NPS and Hadoop

Running the import and export operations from NPS or any other systems is only possible when HDFS is used. If BigInsights uses GPFS, you can only initiate import and export operations from Hadoop.

### Incremental data import when an update or deletion is discovered on NPS

Incremental import to Hadoop is stopped when there was an update or deletion in the table on NPS after the last import, as it leads to data inconsistency. In such situation the import stops with the following error message:

Some records have been deleted or updated after last import. Full import is required to have consistent data. To run incremental import anyway please set `nz.fq.ignore.deleted` to true.

You can use the `nz.fq.ignore.deleted` flag to ignore such error messages and import data anyway.

### Import of some object types is not supported

The following object types are not supported by IBM Fluid Query because they contain unsupported, NPS-specific data types (special columns), such as **rowid**, **datasliceid**, **createxid**, and **deletexid**.

- External table
- Row-secure table (RST)
- Views
- Mviews

Import of these objects will result in an error similar to the following one: ERROR: Column reference "DATASLICEID" not supported for external table.



## Import of some data types is not supported

The following data types are not supported in this release of IBM Fluid Query:

- **varbinary**
- **st\_geometry**

Import of a table that contains these column types will result in the following error: Can't recognize NPS data type

## When importing a clustered base table (CBT), the organizing keys are not retained

When you import a CBT from NPS to Hadoop, the organizing keys are not preserved in the metadata. As a result, when you transfer the table back to NPS, it is exported as a regular table.

You can perform the following steps as a workaround for this limitation:

1. Import a CBT to Hadoop.
2. Before you export it, go to your NPS system and create a table with organizing keys, similar to the schema of the table that you imported to Hadoop.
3. Export only the data from the CBT that is located on Hadoop to the existing table on the NPS side. To do this, you must provide the path to the table directory on HDFS in the export configuration file.

## Export of the Interval data type is not supported on older NPS systems

You cannot export a table that contains **Interval** data types to a Netezza system that runs NPS release older than 7.1.

For these older versions of NPS, you can still import a table containing **Interval** data types to Hadoop but it will be impossible to export such a table back to NPS later.

## IBM Fluid Query does not support complex data types that occur in Hive

The following complex data types are not supported:

- **ARRAY**
- **MAP**
- **STRUCT**
- **UNIONTYPE**
- **BINARY**

## Cannot import tables with the same table name to the same HDFS directory

This problem might occur when importing a table from NPS with the same name to Hadoop. The HDFS directories are not case-sensitive so you cannot import tables with the same table name to the same HDFS directory even if the letter case of table names is different or tables exist in different databases. Both cases will cause data overwriting on HDFS.

To avoid overwriting of data for such tables, change the HDFS directory using the **nz.fq.output.path** parameter in the import XML configuration file. Do this even in the case when the tables are in different databases or their letter case is different. The following example explains this workaround in detail:

1. You have two tables with the same name but different letter case in the same database on NPS, for example tables **test** and **TEST**.
2. In the XML configuration file, set the property for the HDFS directory:

```
<property>
  <name>nz.fq.output.path</name>
  <value>/nzbackup/backup1</value>
</property>
```

3. Run the import for table **test**.
4. Now, in the XML configuration file, set the property for the HDFS directory:

```
<property>
  <name>nz.fq.output.path</name>
  <value>/nzbackup/backup2</value>
</property>
```

5. Run the import for table **TEST**.
6. If you follow this procedure, data is not overwritten and the tables are stored in different locations on HDFS.

## Compatibility of Netezza driver with Java 1.6 and Cloudera

The data movement installation package is delivered with Netezza drivers supporting Java 1.6 or Java 1.7. You can find them by unpacking `fluid-query-import-export.tar` package. The `nzjdbc3.jar` driver supports Java 1.7, and `nzjdbc3_java16.jar` supports Java 1.6. When installing the data movement feature, the installer automatically recognizes the version of Java and Cloudera. When the installed version of Java is 1.6, then `nzjdbc3_java16.jar` is installed, otherwise `nzjdbc3.jar` is installed.

To replace the driver manually:

1. On your system, locate the IBM Fluid Query installation package (minimum version 1.7.0.0), or download the latest package from IBM Fix Central.
2. Go to the directory with the installation package.
3. Unpack `nz-fluidquery-.tar.gz`. Execute the command:

```
gunzip nz-fluidquery-VERSION.tar.gz
```

where *VERSION* is the version of IBM Fluid Query. For example:

```
gunzip /myfolder/nz-fluidquery-v1.7.tar.gz
```

File `nz-fluidquery-VERSION.tar` is created in the directory where the installation package exists.

4. Unpack `nz-fluidquery-VERSION.tar` by executing the command:

```
tar -xvf ./nz-fluidquery-VERSION.tar
```

For example:

```
tar -xvf ./nz-fluidquery-v1.7.tar
```

File `fluid-query-import-export-VERSION.tar` is created in the directory where the installation package exists.

5. Unpack fluid-query-import-export-*VERSION*.tar by executing the command:  

```
tar -xvf ./fluid-query-import-export-VERSION.tar
```

For example:

```
tar -xvf ./fluid-query-import-export-v1.7.0.0.tar
```

Two files are created:

- nzjdbc3.jar
  - nzjdbc\_java16.jar
6. To manually install the Netezza driver supporting Java 1.7, copy nzjdbc3.jar to the data movement installation directory. Run the command:  

```
cp ./nzjdbc3.jar /fluidquerylocal
```
  7. To manually install the Netezza driver supporting Java 1.6, copy nzjdbc3\_java16.jar to the data movement installation directory. Run the command:  

```
cp ./nzjdbc3_java16.jar /fluidquerylocal/nzjdbc3.jar
```

## Limitations on VARCHAR data types

By default, the parameter **fq.hive.usevarchar** is set to false, that is, the VARCHAR data type is changed to STRING on Hive. This is caused by a number of limitations when converting to VARCHAR on Hive, for example:

- On Cloudera, the escape character ('\') is not processed correctly.
- Export in binary mode does not process characters with codes 128-255 in varchar column.
- In BigInsights 3 there is a problem with synchronizing Hive table metadata with BigSQL: Error Column "COL\_CHAR", type "varchar(1)" is not supported

## Handling a field delimiter and end of line character in data movement

Hive does not support new line and end of line characters within data. Records including these characters are separated, thus data is handled incorrectly. When querying any data format other than binary (for example text, Avro, Parquet), the new line and end of line characters cannot be handled correctly.

You can use compression and store data including these characters on Hadoop in Netezza compressed (binary) mode. Then, when querying data, newline and carriage return characters are replaced with a space (' '). This is performed by the nzetc tool on the fly, during decompression from the binary format. When you export data back to NPS, the data is not affected. The solution does not work for mixed mode of transfer.

## Limitation on TIMESTAMP data type for BigSQL

Date range in BigSQL starts from 1400-01-01. When importing a table to BigSQL with a TIMESTAMP column that contains dates earlier than year 1400, BigSQL shows NULL value for these dates.

## Limitation on DATE and TIMESTAMP containing BC (Before Christ)

Moving data with columns DATE or TIMESTAMP that contain BC (Before Christ), for example ('1400-01-13 BC', '1400-01-13 08:13:33 BC'), might fail with error, as such date format is not supported:

- Hive does not support BC format as described in <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Types#LanguageManualTypes-date>
- NPS does not support unload for BC format. For more information see [https://www.ibm.com/support/knowledgecenter/en/SSULQD\\_7.2.1/com.ibm.nz.load.doc/r\\_load\\_datestyle.html](https://www.ibm.com/support/knowledgecenter/en/SSULQD_7.2.1/com.ibm.nz.load.doc/r_load_datestyle.html)

## Errors when exporting a Hive table with name or column name starting with underscore

Exporting a table with name or column name starting with \_ (underscore) is not supported. As a workaround, you can create a view based on this table, but with name or column name not starting with underscore.

## Troubleshooting import and export of Hadoop formats

Read how to solve problems when importing and exporting Hadoop specific formats.

### Known limitations

Following is a list of limitations and possible errors when using IBM Fluid Query with Hadoop formats. These limitations were found when testing the data movement feature on the listed environments at the time of 1.7 release. Note that these issues might be fixed on the side of database service provider at any time.

- Import to Hadoop formats is implemented as a Hive statement which inserts text data transferred from NPS into a Hive table, that is stored in one of the Hadoop formats. The same applies to export in the opposite direction. Therefore, data transfer strongly depends on Hive and on the implementation of formats on a particular Hadoop distribution. This means that any limitations and defects that exist on a particular Hadoop distribution also affect IBM Fluid Query data movement.
- The `enable.doAs=true` setting in Hive is required in order to import to Hadoop format. With this setting operations executed in Hive are executed with a connected user, rather than the user who started HiveServer.
- Nonprinting control characters cannot be processed when using Hadoop formats. If your data contains these characters, do not import to Hadoop formats.
- Limitation on importing data containing national characters to Hadoop formats:  
A problem with national characters may occur when the data on NPS is stored in CHAR/VARCHAR columns. To avoid errors, store data with national characters in NCHAR/NVARCHAR columns. In this way, national characters for both imported and exported data will be processed correctly.
- Do not use Avro format when your data contains NUMERIC fields in NPS, which are then mapped to DECIMAL on Hadoop. Converted data might be inconsistent.
- Moving a table containing timestamp data type that is stored in ORC format might lead to data inconsistencies. This problem depends on JDBC driver and

Hive version. You should always double-check that the data is consistent after movement. You can use checksum calculation for that purpose.

- IBM BigInsights limitations:
  - On BigInsights 3, importing to Avro and Parquet is not supported and it results in the following error:  
`com.ibm.nz.fq.common.exception.hive.HiveException: Unable to execute Hive sql. Error while processing statement: FAILED: SemanticException Unrecognized file format in STORED AS clause: avr`
  - On BigInsights 3, do not import tables to ORC format if they contain bigint and int8 data types.
  - On BigInsights 4.0, using Parquet is not supported and it results in one of the following errors:  
`Incompatible Long vector column and primitive category VOID`  
`Unimplemented vector assigner for writable type class`  
`org.apache.hadoop.hive.serde2.io.HiveDecimalWritable`
  - On BigInsights 4.0, importing timestamp data type to Avro format is not supported.
  - Import to BigSQL is not supported. You can import a table to Hive and then synchronize it manually to BigSQL. However, querying files imported in Avro format from BigSQL is not supported even after manual synchronization.
- Cloudera limitations:
  - Import to Hadoop formats is not supported for Cloudera versions prior to 5.3.
  - Import of tables with new line characters within rows is not possible for Parquet. The error symptom is extra rows with unaligned data on the record with new line character. You can import such data using other compression formats.
  - Import of timestamp data type to Avro format is not supported on Cloudera.
  - Impala does not support ORC format.
  - Querying imported Avro tables in Impala is not supported on Cloudera versions prior to 5.5.
- Hortonworks limitations:
  - On Hortonworks 2.2, it is not possible to specify compression type in `nz.fq.output.compressed` for Avro. You can leave the parameter empty and then default Hadoop compression is used.
  - On Hortonworks 2.2 import to Parquet is not supported.
  - When using Hortonworks 2.3, the `add jar` command for Hive JDBC connection might result in `NullPointerException`

## Common errors and solutions

- Data movement fails with the following error:  
`ERROR com.ibm.nz.fq.ConnectionValidator - Unable to execute Hive sql. Error while processing statement: Cannot modify avro.output.codec at runtime. It is not in list of params that are allowed to be modified at runtime`  
Add or modify a custom Hive parameter  
`"hive.security.authorization.sqlstd.confwhitelist.append"` with value `avro.output.codec` as in the following example:  

```
<property>  
<name>hive.security.authorization.sqlstd.confwhitelist.append</name>  
<value>avro.output.codec</value>  
</property>
```

If multiple entries are needed, for instance, both Parquet and Avro compression is used, the parameter should be specified as follows:

```
<property>
<name>hive.security.authorization.sqlstd.confwhitelist.append</name>
<value>parquet\compression|avro\output\codec</value>
</property>
```

- Export of Hive table in Parquet fails if you use WHERE clause on a FLOAT column. This is a Hive issue and it is described in the following topic:  
<https://issues.apache.org/jira/browse/HIVE-13114>

## Symptoms of failed import to Hadoop formats

When the import operation to Hadoop formats is interrupted for any reason, two tables instead of one are created in Hive/HDFS:

### Destination table `<table_name>_npscopy`

- For AVRO format, any query might throw the following error:  

```
java.io.IOException: org.apache.hadoop.hive.serde2.avro.AvroSerdeException:
Neither avro.schema.literal nor avro.schema.url specified,
can't determine table schema
```
- For other formats table behaves like an empty table.

### Interim table

`<table_name>_<process_id>_<milliseconds_from_1970.01.01>_npscopy`

for example: `table1_3480_1487843174319_npscopy`

- User is not able to query this table.
- Error: `java.io.IOException: cannot find class com.ibm.nz.codec.NzCopyInputFormat`

If two tables are created after the import operation, and you see any of these error messages when querying tables, it means that the import operation failed at some point.

## Custom settings for Hive

You can define custom settings before data is inserted into Hive by setting **nz.fq.data.format.setting** property in the configuration XML file. It allows you to define your own properties for compression, or any additional Hive settings related to the destination format. As a value, provide a list of comma-separated settings (without the SET command):

```
<property>
  <name>nz.fq.data.format.setting</name>
  <value>hive.exec.compress.output=true,avro.output.codec=snappy</value>
</property>
```

As a result, a list of the following commands is sent to Hive:

```
SET hive.exec.compress.output=true;
SET avro.output.codec=snappy;
```

## Verifying intermediate text data files

If the results of import to Hadoop formats are incorrect, you can analyze the problem with the help of **nz.fq.debug.keep.tmp** parameter. When set to true, the intermediate text files that are used for insert to the target table in Hadoop format, are not deleted after the import is finished. This parameter can only be used with **fq.data.directmovement=false**, which is default. After importing with **nz.fq.debug.keep.tmp=true**, text files are kept in `<target_table_dir>/tmp` subdirectory and Hive text table based on this directory called `<target_table>NPSCOPY` is also available.

---

## Troubleshooting connection problems

If you encounter problems running the **fqConfigure.sh** script to define connections to service providers, there are some suggested troubleshooting steps.

- If the **fqConfigure.sh** script returns a Java error such as `java.lang.NoClassDefFoundError: org.apache.<class name>`, you might be missing a Java JAR file that is required for the connection to the Hadoop service provider. Review the class name shown in the error message, and see “JDBC drivers for predefined data connectors” on page 3-2 for more information about the JDBC files required for each connection.
- If the **fqConfigure.sh** script returns an error similar to Execution failed due to a distribution protocol error that caused deallocation of the conversation. A DRDA Data Stream Syntax Error was detected. Reason: 0x3. ERRORCODE=-4499, SQLSTATE=58009, check your port number specified in the command. If you did not specify a `--port` value, your Hadoop service provider may not be using the default port. If you specified a `--port` value, you might have specified an incorrect value.

- For problems with Kerberos:

If you use Kerberos authentication, make sure that the system time on your Netezza appliance matches the system time for your Kerberos realm.

If you have problems using Kerberos authentication to connect to a service provider, try connecting to the Kerberos realm by its domain name (not the IP address). For example, try the following commands:

- `$ ping your.realm.com`
- `$ telnet your.realm.com 88`
- `$ cat /etc/hosts` to check the host definitions for your site

If you have problems connecting to an Impala or Hive server, try the following commands:

- `$ telnet your.impala.com 21050`
- `$ telnet your.hive.com 10000`
- `$ cat /etc/hosts` to check the host definitions for your site

Check that your hostname resolves to your external IP address that the Kerberos server uses and not 127.0.0.1:

- `$ hostname` to display the system hostname
- `$ ping hostname` where *hostname* is the value obtained in the previous command.
- `$ cat /etc/hosts` to check the host definitions for your site

- Connection configuration to Hive on BigInsights fails when using SSL:

1. You might see the following error:

```
javax.net.ssl.SSLKeyException: RSA premaster secret error
```

which is a common Java problem. To solve it, download the Unrestricted SDK JCE policy files for IBM Java available at [https://www-01.ibm.com/marketing/iwm/iwm/web/reg/download.do?source=jcesdk&lang=en\\_US&S\\_PKG=13\\_01&cp=UTF-8](https://www-01.ibm.com/marketing/iwm/iwm/web/reg/download.do?source=jcesdk&lang=en_US&S_PKG=13_01&cp=UTF-8) and update the files on NPS in the following locations:

```
/nz/export/ae/languages/java/java_sdk/host/jre/lib/security/  
local_policy.jar  
  
/nz/export/ae/languages/java/java_sdk/host/jre/lib/security/  
US_export_policy.jar
```

2. When using the predefined Hive connector, you might see the following error:

```
Could not open client transport with JDBC Uri:
jdbc:hive2://hive_server_host_name:10000/default;SSL=true;;
Invalid status 21
```

which is caused by the fact that IBM Fluid Query is generating property `SSL=true` in upper case letters, instead of lower case. To avoid this issue, you can use generic connector which allows you to specify the connection URL on your own:

- a. Create the configuration XML file (the following example uses variables):

```
<?xml version="1.0" encoding="UTF-8"?>
<connection>
  <driverClass>org.apache.hive.jdbc.HiveDriver</driverClass>
  <classPath>/nz/export/ae/products/fluidquery/libs/cloudera/hive</classPath>
  <connectionURL>jdbc:hive2://${HOSTNAME}:${PORT}/default;ssl=true;</connectionURL>
  <jdbc-properties>
    <property>
      <name>user</name>
      <value>${USERNAME}</value>
    </property>
    <property>
      <name>password</name>
      <value>${PASSWORD}</value>
    </property>
  </jdbc-properties>
</connection>
```

- b. When running `fqConfigure`, modify provider and service command arguments to generic, and, using `connectionXmlPath`, specify the path to the XML file that you created:

```
./fqConfigure.sh --host hive_server_host_name --port 10000 --provider generic
--service generic --connectionXmlPath /your/path/to/the/file.xml --config your_file_name
--username your_user --password your_password --ssl
```

---

## Troubleshooting problems with Kerberos

Read about known limitations and common issues when using Kerberos authentication.

### Known limitations:

1. If you use Kerberos authentication, make sure that the system time on your Netezza appliance matches the system time for your Kerberos realm.
2. To use Kerberos, you must use IBM Netezza Analytics with Java 1.7, that is version 3.2.1 or higher, or use Java 1.6 that is shipped with IBM Fluid Query version 1.6.0.1 or earlier. IBM Fluid Query 1.7 does not include any Java package. If you perform a fresh installation of IBM Fluid Query 1.7 on NPS with IBM Netezza Analytics 3.2 or earlier, Kerberos authentication is not supported. Try one of the following workarounds:
  - Install IBM Fluid Query version 1.6 (which includes a Java package that is working with Kerberos) and then upgrade Fluid Query to version 1.7.
  - Upgrade IBM Netezza Analytics to version 3.2.1 or higher.
3. IBM Fluid Query does not support Kerberized IBM BigInsights 4.2.

### Common issues:



## Cannot run data movement from NPS system with Kerberos authentication

If your NPS system uses Kerberos authentication for its database user accounts, note that you cannot run the data movement feature from the NPS host to import or export files. As an alternative, you can run the data movement feature from the Hadoop service provider.

## Services authenticated with Kerberos in remote mode do not work

If you use mixed authentication (Kerberos and user/password) in remote mode:

1. You must restart `fqRemote.sh` after each registration with Kerberos.
2. If you have any remote Kerberos connections, you might break them if you restart `fqRemote.sh` after a non-Kerberos remote registration. To fix the situation, register remote with Kerberos again and restart `fqRemote.sh`.

If you use different `krb5.conf` files during SQL connector configuration, only the last one registered before starting or restarting `fqRemote.sh` is valid. You must either have your systems configured in such a way that they use only one `krb5.conf` file, or you have to re-register some connectors and restart `fqRemote.sh`.

## `fqConfigure` fails with Kerberos error: return code 14 "KDC has no support for encryption type"

To solve the problem, add `default_tkt_enctypes` entry to `[libdefaults]` section of Kerberos configuration file. The `default_tkt_enctypes` lists combinations of the encryption type, such as checksum, available for service tickets encryption and hashing. Content may vary, for example:

```
default_tkt_enctypes = aes128-cts-hmac-sha1-96
```

## kinit problems

Symptoms:

`fqConfigure` fails with error Unable to authenticate with Kerberos

Troubleshooting steps:

1. Try connector configuration using both possible methods:
  - with keytab **--keytab**
  - with password, using **--password** or password prompt
2. Check whether the system embedded `kinit` works using keytab and password:

```
kinit <myprinc> -k -t <ktab_file>
kinit <myprinc>
```

For troubleshooting, you can set the `KRB5_TRACE` variable:

```
export KRB5_TRACE=/tmp/kinit_trace
```

and then, after running `kinit`, you can check trace in `/tmp/kinit_trace`.

3. Check whether the `kinit` from Java works using keytab and password:

```
/nz/export/ae/languages/java/java_sdk/host/jre/bin/kinit <myprinc> -k -t <ktab_file>
/nz/export/ae/languages/java/java_sdk/host/jre/bin/kinit <myprinc>
```

IBM Fluid Query uses Java kinit, so if manual kinit with Java does not work, there is no point investigating IBM Fluid Query.

Symptoms:

- Error: Unable to authenticate with Kerberos
- The following information is in the log file:

```
Caused by: com.ibm.security.krb5.KrbException, status code: 25
message: Additional pre-authentication required
Caused by: com.ibm.security.krb5.Asn1Exception, status code: 906
message: Unexpected ASN1 identifier
```

- kinit executed from Java fails, system kinit works fine.
- Keytab encryption is arcfour-hmac, for instance:

```
[nz@testnz /]$ klist -k -e -t /nz/export/ae/products/fluidquery/myprinc.keytab
Keytab name: FILE:/nz/export/ae/products/fluidquery/myprinc.keytab
KVNO Timestamp Principal
-----
1 01/16/2016 13:01:25 myprinc@mydomain.com(arcfour-hmac)
```

Possible reason:

Problem may be caused by arcfour-hmac-md5 encryption specified in krb5.conf. This entry may cause problems in Java, but not in system kinit. As a workaround, arcfour-hmac-md5 needs to be replaced with arcfour-hmac in krb5.conf file. For more information see <http://www-01.ibm.com/support/docview.wss?uid=swg1IV78190>.

Symptoms:

- Java kinit works fine
- IBM Fluid Query log contains entry New ticket is stored in cache file

Troubleshooting steps:

You can use the kvno command to check whether service ticket for a particular service can be obtained:

```
kinit <client principal> (-k -t <keytab file>)
kvno <service principal>
klist
```

---

## IBM Fluid Query log files

Read about the log files for data connector, data movement, and how to work with a log collector.

### Data connector log files

Review the data connector log files for information that might be helpful for feature usage and troubleshooting tasks.

After you install the data connector feature, you can obtain log files for various commands, tasks, and actions in the `/nz/export/ae/products/fluidquery/logs` directory. The log files can be useful references for assistance with troubleshooting configuration problems or other aspects of the product operation.

If you use the `--debug` option when you register the data connector functions using the **fqRegister.sh** script, the software creates log files each time a query runs and calls the functions. After you finish debugging the functions and your queries are operating as expected, make sure that you re-register the functions without the `--debug` option to stop the log files created with each query.

Be sure to check the `/nz/export/ae/products/fluidquery/logs` directory periodically and delete old log files that you no longer need. This can help to keep the `/nz` directory area from filling and possibly impacting NPS operations.

## Data movement log files

For troubleshooting purposes, you can configure logging for the data movement feature. You can also review mapper logs when import or export fails.

### About this task

When the data import or export operation fails, you might refer to the following log files for mappers:

- on Hadoop: `/fluidqueryLocal/var/log/mapper`
- on Netezza: `/nz/export/ae/products/fluidquery/logs/mapper`

You can also use the following procedure to save additional log file when running data movement.

### Procedure

1. Create a `log4j.properties` file in the directory where you start the import and export operations.
2. Paste the following code in the file:

```
# initialize root logger with level ERROR for stdout and fout
log4j.rootLogger=DEBUG,file
log4j.logger.com.ibm.nz=DEBUG,stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.Threshold=INFO
log4j.appender.stdout.layout.ConversionPattern=%d %p [Thread-%t] %c{2}: %m%n

# add a FileAppender to the logger file
log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.File=fq.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d %p [Thread-%t] %c{2}: %m%n
```

### Results

Logs will be produced in a `fq.log` file in the same directory.

## Working with log collector

You can use the `fqLogCollector.sh` script to gather information that might be valuable to the IBM support team when troubleshooting.

## About this task

The `fqLogCollector.sh` script is located in `/nz/export/ae/products/fluidquery/fqSupportTools` on NPS, and in `/fluidqueryLocal/scripts/fqSupportTools` on Hadoop.

The log collector produces the output package `fqlog_<date>_<time>.tgz` in the `fqSupportTools` folder. You deliver this package to the IBM support team.

For command reference on `fqLogCollector.sh`, see “The `fqLogCollector` script” on page 6-16.

---

## Chapter 6. Command and function reference

This section contains the descriptions of the functions and commands used with the IBM Fluid Query feature.

---

### The fqConfigure script

You use the **fqConfigure.sh** script to configure connections from NPS to the service providers.

#### Syntax

The **fqConfigure.sh** script has the following syntax.

```
fqConfigure.sh --host hostname [--port number ]  
--provider name --service service_name  
[options] [authentication]
```

#### Inputs

The **fqConfigure.sh** script takes the following inputs:

*Table 6-1. The fqConfigure.sh input options*

Input	Description
--host	Specifies the host name of the system where the service is running.
--port	<p>Specifies the port number for the service to which you are connecting. The command uses the following default port numbers for the services:</p> <ul style="list-style-type: none"><li>• For Hive, 10000</li><li>• For SparkSql: 10000</li><li>• For Impala, 21050</li><li>• For BigSQL, 51000</li><li>• For BigSQL v1, 7052</li><li>• For DB2, 50000</li><li>• For PureData System for Operational Analytics, 50000</li><li>• For dashDB, 50000</li><li>• For Oracle, 1521</li><li>• For IBM PureData System for Analytics, 5480</li></ul> <p><b>Note:</b> If the service administrator started the service on a custom port, use the --port option to specify the port for your service environment.</p>

Table 6-1. The **fqConfigure.sh** input options (continued)

Input	Description
<code>--database database_name</code>	Specifies the name of the database to which you are connecting. This option is required when establishing a connection to DB2, PureData System for Operational Analytics, dashDB, IBM PureData System for Analytics, and Oracle. For the other services, <code>--database</code> is optional.
<code>--provider provider_name</code>	Specifies the vendor name of the service provider. You can specify one of the following values: <ul style="list-style-type: none"> <li>• IBM</li> <li>• Cloudera</li> <li>• Horton</li> <li>• Oracle</li> <li>• Generic</li> </ul>
<code>--service service_name</code>	Specifies the type of service to which you are connecting. You can specify one of the following values: <ul style="list-style-type: none"> <li>• Impala</li> <li>• BigSql</li> <li>• BigSqlv1</li> <li>• Hive</li> <li>• SparkSql</li> <li>• Oracle</li> <li>• DB2</li> <li>• dashDB</li> <li>• PDOA (for PureData System for Operational Analytics)</li> <li>• PDA (for IBM PureData System for Analytics)</li> <li>• FQDM (for the data movement support when run from NPS systems)</li> <li>• Generic</li> </ul>
<code>--config file_name</code>	Specifies the name of the configuration file name that is created after the connection is successful. If the configuration name is "MyConf" then the system creates a <code>MyConf.properties</code> file. The default configuration name is default.  The command displays a message if the config file already exists, and you can choose whether to overwrite the file. If you choose not to overwrite the file, the command exits and you can run the command with a different file name.

Table 6-1. The `fqConfigure.sh` input options (continued)

Input	Description
<code>--driver-path <i>path_name</i></code>	Specifies the pathname of the driver files to use for the configuration. For environments that use more than one version of the same service provider, you can use this option to specify the non-default location of a set of driver files. For example, if you have a Cloudera version 4.7 service provider and a Cloudera version 5.3 service provider, you place one version of the drivers in the default directory <code>/nz/export/ae/products/fluidquery/libs/cloudera/hive</code> . The drivers for the second version must be in a custom directory that you create, and you specify the location using the <code>--driver-path</code> option.
<code>--varchar-size <i>bytes</i></code>	Specifies the default VARCHAR size to use when converting from String data type to Netezza VARCHAR type during FqRead operations. The default value is 1000. The maximum value for a VARCHAR is 64,000. See the FqRead section for more information. (Only used for JDBC-related services.)
<code>--str-to-nvarchar</code>	If you use this argument, all string types like VARCHAR, VARCHAR2, STRING and similar, are mapped to NVARCHAR type. (By default, all string types are mapped to VARCHAR, but it does not support unicode characters.)
<code>--fqdm-conf <i>path</i></code>	For the IBM Fluid Query data movement (FQDM) feature, specifies the path to the FQDM configuration file. If you are creating a connection to allow your NPS users to run functions that import files to or export files from a service provider, you use this option to configure the connection. This option is required when you specify <code>--service FQDM</code> to use the FQDM feature. A template file is available at <code>conf/fq-remote-conf.xml</code> .
<code>--version</code>	Displays the version and build information for the data connector utilities.
<code>--help</code>	Displays usage and syntax for the command.
<code>--debug</code>	Writes more information about the processing of the script to the log file for troubleshooting purposes.
<code>--no-validation</code>	Blocks validation of configuration.

## Generic connector options

Table 6-2. Generic connector input options

Input	Description
--connectionXmlPath	Specifies path to the XML file that defines connection configuration for generic connector.
--dataTypesXmlPath	Specifies path to the XML file defining data types mapping for generic connector. This parameter is optional.

## Authentication options

For each service connection, you can specify SSL authentication, user account and password authentication, or Kerberos authentication. You cannot specify both the user/password and Kerberos options, or both the SSL and Kerberos options.

Table 6-3. SSL authentication options

Input	Description
--ssl	If your configuration uses SSL authentication, specify this option to note that SSL is required to connect to the target service.
--ssl-truststore <i>path name</i>	Specifies the Full path to SSL truststore that should be used for the secure connection. If not provided, a new one is generated.
--ssl-truststore-password <i>password</i>	Specifies the custom truststore for authentication. If you specify a custom truststore, you must also provide a password.

Table 6-4. User and password authentication options

Input	Description
--username <i>name</i>	Specifies the user account name to use during authentication.
--password <i>password</i>	Specifies the password for the local user name or SSL authenticated user name. You can also skip this parameter and you are automatically prompted for password.
--autoGenerateKey	Automatically generates encryption key for encrypting passwords stored in the connector configuration file.
--KeyFileOutfilepath	Specifies the location of the generated encryption key, other than default.
--KeyFilefilepath	Specifies the user-provided encryption key file.



Table 6-5. Kerberos authentication options

Input	Description
<code>--service-principal <i>name</i></code>	Specifies the target service principal. This value must be the same user principal that was used when starting the service. (For example: <code>impala/myhost.example.com@example.com</code> )
<code>--client-principal <i>name</i></code>	Specifies the Kerberos client principal.
<code>--krb5-conf <i>path name</i></code>	Specifies the full path to the Kerberos configuration file, as described in “Kerberos configuration file” on page 3-9. If you do not specify a <code>--krb5-conf</code> option, the command uses the default value <code>/etc/krb5.conf</code> .
<code>--password <i>password</i></code>	Specifies the password for the client-principal. The <code>--password</code> and <code>--keytab</code> options have the following behaviors: <ul style="list-style-type: none"> <li>• If you specify <code>--keytab</code> but not <code>--password</code>, the command uses the specified keytab file.</li> <li>• If you specify <code>--keytab</code> and <code>--password</code>, the command uses the specified keytab file and ignores the password.</li> <li>• If you specify <code>--password</code> but not <code>--keytab</code>, the command creates the default keytab.</li> <li>• If you do not specify <code>--password</code> or <code>--keytab</code>, the command prompts you for a password.</li> </ul>
<code>--keytab <i>path name</i></code>	Specifies the full path to the keytab that you are using. For more information about the <code>--keytab</code> and <code>--password</code> processing, see the <code>--password</code> description.

Table 6-6. No authentication options

Input	Description
<code>--no-auth</code>	For Impala service only, specifies that no authentication ( <code>auth=noSasl</code> ) should be used for the connection.

## Description

You use the **fqConfigure.sh** command to create a connection to a service provider. When the configuration is successful, the command displays the message `Connection configuration success`. If the configuration fails, the command displays an error message. For more information, review the log file in the `/nz/export/ae/products/fluidquery/logs` directory.

If you want to create more than one configuration, use the `--config` option, which creates a configuration file using the name that is provided as a parameter. You can use the configuration file later when registering the function.

In most cases, you use the command to configure connections to service providers for the data connection operations. You can also use the command to create FQDM connections to Hadoop service providers if users will be running import and export operations from the NPS appliance.

## Usage

The following sample commands show some of the common command uses and syntax. The IP addresses and domains shown below are samples and must be replaced with the specifics in your specific service provider environment.

- To configure a connection to Cloudera/Impala using Kerberos authentication:  

```
[nz@nzhost-h1 ~]$ ./fqConfigure.sh --host 192.0.2.1
--port 21050 --service-principal impala/myhost.example.com@example.com
--client-principal myuser@example.com --krb5-conf /mypath/krb5.conf
--config myConfig --provider cloudera --service impala
```
- To configure a connection to IBM BigInsights with user and password authentication:  

```
[nz@nzhost-h1 ~]$ ./fqConfigure.sh --host 192.0.2.1
--port 7052 --username biadmin --config biBigSql --provider ibm
--service BIGSQL
Please enter your client password:
```
- To display the version information for the data connector utilities:  

```
[nz@nzhost-h1 ~]$ ./fqConfigure.sh --version
FluidQuery Version: 1.5.0.0 [Build 150630-186]
```
- To configure a connection with generic connector:  

```
/nz/exp/nz/export/ae/products/fluidquery/fqConfigure.sh --provider generic
--service generic --connectionXmlPath /nz/export/ae/products/fluidquery/netezza.xml
--config sample_generic --username admin
--dataTypesXmlPath /nz/export/ae/products/fluidquery/types.xml
```

---

## The fqDownload script

You use the **fqDownload.sh** script to download the Hadoop libraries, Hive libraries, and client configuration files from the target Hadoop service provider for use with the data movement feature run from NPS appliances.

### Syntax

The **fqDownload.sh** script has the following syntax.

```
fqDownload.sh --host hostname --user username
--service [BIGSQL|HIVE] [--path directory]
```

### Inputs

The **fqDownload.sh** script takes the following inputs:

*Table 6-7. The fqDownload.sh input options*

Input	Description
<code>--host <i>hostname</i></code>	Specifies the host name or IP address of the server where the Hadoop service provider is running.
<code>--user <i>username</i></code>	Specifies the user name with which to log in to the specified Hadoop server. The command prompts you for the password for the specified user account.

Table 6-7. The **fqDownload.sh** input options (continued)

Input	Description
<code>--service [BIGSQL HIVE]</code>	Specifies the service that you use to transfer data. It downloads only the drivers that you need. There are two supported types: Hive and BigSQL.
<code>--path <i>pathname</i></code>	Specifies the directory on the Netezza host to which all the files will be downloaded. If you do not specify this option, the download process uses the default driver location for the service provider that you are accessing. For example, if you are accessing an IBM provider, the files are stored in the <code>/nz/export/ae/products/fluidquery/libs/ibm/fqdm</code> directory.

## Description

You use the **fqDownload.sh** command to load the files that are required for an NPS appliance to run the IBM Fluid Query data movement feature. The command uses the `ssh` and `scp` commands to connect to the host, identify the Hadoop provider and its software version, and then downloads all the required files.

This command requires the service provider to be installed in its default locations. The script prompts for but does not store the user password that you use to access the host. The script prompts for the password twice: once for version discovery and another time to download the service provider files. If you are downloading files from multiple versions of the same service provider, you must use the `--path` option to specify the location on the NPS host to store any subsequent sets of service provider files.

## Usage

The following provides a sample command session. The command uses the sample IP 1.2.3.4 for the service, and the full list of files are intentionally truncated for the example.

```
[nz@nzhost fluidquery]$ ./fqDownload.sh --host 1.2.3.4 --user root --service HIVE
#### Checking if system is an NPS Host...

System is an NPS appliance.

#### Checking hadoop version...

root@1.2.3.4's password:
Hadoop 2.6.0-IBM-7
Subversion git@biосс.svl.ibm.com:bigtop -r bc50d47c60cf8f1b69560a224a3aed0227dfcd1d
Compiled by jenkins on 2015-03-28T04:34Z
Compiled with protoc 2.5.0
From source with checksum d72cfbc86823581782f5576bafef7f69
This command was run using /usr/iop/4.0.0/hadoop/hadoop-common-2.6.0-IBM-7.jar

Provider: IBM
Hadoop full version: 2.6.0-IBM-7
Hadoop version: 2.6.0

#### Downloading Hadoop libraries and client config files into
/nz/export/ae/products/fluidquery/libs/ibm/fqdm

Downloading from 1.2.3.4:
/usr/iop/current/hadoop-client/client/
/usr/iop/current/hive-client/lib/
/etc/hadoop/conf/

Running command: scp root@9.158.143.116: "/usr/iop/current/hadoop-client/client/*.jar
/usr/iop/current/hive-client/lib/*.jar /etc/hadoop/conf/*.xml "
/nz/export/ae/products/fluidquery/libs/ibm/fqdm

root@1.2.3.4's password:
activation-1.1.jar
activation.jar
apacheds-18n-2.0.0-M15.jar
apacheds-18n.jar
apacheds-kerberos-codec-2.0.0-M15.jar
apacheds-kerberos-codec.jar
api-asn1-api-1.0.0-M20.jar
..

#### Checking the NPS jdbc library...

Copying nzjdbc3.jar into the /nz/export/ae/products/fluidquery/libs/ibm/fqdm
```

## The fqRegister script

You use the **fqRegister.sh** script to register the data connector functions or the data movement functions in a target Netezza database.

### Syntax

The **fqRegister.sh** script has the following syntax.

```
fqRegister.sh [--user username] [--pw password]
[--db name] [--udtf name]
[--config name] [--remote] [--xmx] [--xms]
[--unregister] [--help] [--debug]
```

## Inputs

The **fqRegister.sh** script takes the following inputs:

Table 6-8. The **fqRegister.sh** input options

Input	Description
<code>--user name</code>	Specifies the database user account to access the Netezza database and register the functions. The default is the value of the <code>NZ_USER</code> variable.
<code>--pw password</code>	Specifies the password for the database user account. The default is the value of the <code>NZ_PASSWORD</code> variable.
<code>--db name</code>	Specifies the Netezza database name where function is registered. The default is the value of the <code>NZ_DATABASE</code> variable. <b>Note:</b> You can only use database name that includes quotation marks when using IBM Netezza Analytics 3.2.1 or above. You must also escape double quotes in the <code>--db</code> parameter, as in <code>\ "dbname\"</code> . Quotation marks are required to register a database with a name that is written in camel case.
<code>--udtf name</code>	Specifies the name of the user-defined table function that you are registering. The default name is <code>FqRead</code> for JDBC-related services.
<code>--config name</code>	Specifies the name of the configuration file that the function uses to connect to the Hadoop service. This is the config file created by the <b>fqConfigure.sh</b> script.  When specifying a value, do not include the <code>.properties</code> suffix of the filename. The default configuration name is <code>default</code> .
<code>--overwrite</code>	Specifies that you want to overwrite any functions that are already registered in the database with the same name. Otherwise, you are prompted to replace the existing functions.
<code>--remote</code>	Specifies that the function is registered in REMOTE mode. By default all functions are registered in LOCAL mode. For more details about LOCAL/REMOTE mode, see “About local and remote mode functions” on page 3-21.
<code>--xmx value</code>	Specifies the Java <code>xmx</code> property used by Netezza Analytics when running the registered function. The default value is <code>-Xmx1024M</code> .
<code>--xms value</code>	Specifies the Java <code>xms</code> property used by Netezza Analytics when running registered function. The default value is <code>-Xms256M</code> .
<code>--unregister</code>	Specifies that you want to unregister, or remove, the function from the specified database.

Table 6-8. The **fqRegister.sh** input options (continued)

Input	Description
<b>--help</b>	Displays usage and syntax for the command.
<b>--debug</b>	Writes more information about the processing of the script to the log file for troubleshooting purposes. If you use the <b>--debug</b> option when you register the functions, the software creates log files each time a query runs and calls the functions. After you finish debugging the functions and your queries are operating as expected, make sure that you re-register the functions without the <b>--debug</b> option to stop the log files created with each query.
<b>--addJavaParams</b>	Allows to provide java parameters for specific UDTF functions.

## Description

You use the **fqRegister.sh** command to add or register the IBM Fluid Query-related functions for use in a Netezza database. You use the command to register the data connector functions or the data movement functions as needed for your Netezza users.

When the configuration is successful, the command displays the message Functions and credentials are successfully registered in database "*dbname*". If the configuration fails, the command displays an error message. For more information, review the log file in the `/nz/export/ae/products/fluidquery/logs` directory.

During a successful registration, the command updates the specified configuration file with a reference to the function name. If you want to unregister (or remove) that function at a later time, you can use the **--unregister** option to remove the function from the database. When you unregister, if you do not specify a function name with the **--udtf** option, the command removes the last function that was registered for that configuration file. As a best practice, when you are unregistering a function, use the **--udtf** option to specify which function you want to unregister/remove. If the unregister operation fails with the error Unregistering failed for database "MYDB" then retry the command and make sure that the database and function name (**--udtf**) for the command are correct.

## Usage

The following provides some of the command uses and sample syntax:

- To register the function using the default setup:

```
[nz@nzhost-h1 ~]$ ./fqRegister.sh
```

This command registers the functions named FqRead and creates a default.properties configuration file. The default `-Xmx1024M` and `-Xms256M` mode settings are also used.

- To register the functions in a specific database:

```
[nz@nzhost-h1 ~]$ ./fqRegister.sh --config MyConfig --db MyDb
```

This command registers the functions named FqRead in the MyDb database and creates a MyConfig.properties configuration file. The default -Xmx1024M and -Xms256M mode settings are also used.

- To register the functions using a more advanced form of the command:

```
[nz@nzhost-h1 ~]$ ./fqRegister.sh --config MyConfig
--udtf MYFUNCTION --xmx -Xmx512M --xms -Xms128M --remote
```

This command registers the NPS function named MYFUNCTION in the NZ\_DATABASE database and creates a MyConfig.properties configuration file. The xmx is set to 512M and xms is set to 128M, and the function is defined as a remote (not local) executable.

- To register the data movement functions on a Netezza host to perform imports and exports using function calls:

```
[nz@nzhost-h1 ~]$ ./fqRegister.sh --config fqdm
--udtf toMyHadoop,fromMyHadoop
```

This command registers two functions using FQDM configuration: toMyHadoop() and fromMyHadoop(). Use the toMyHadoop function to import tables to the Hadoop destination system. Use the fromMyHadoop to export tables from the Hadoop server to the NPS host. The default function names are toHadoop() and fromHadoop(). See the section “The **toHadoop** and **fromHadoop** functions” on page 6-14 for more information.

- To register a function providing additional Java parameters:

```
[nz@nzhost-h1 ~]$ ./fqRegister.sh --addJavaParams -Ddb2.jcc.charsetDecoderEncoder=3 --debug
--config IFQ_BI --udtf toQuote,fromQuote --db \"TEST_quotes\"
```

- To unregister a function from the database, which removes the function for use:

```
[nz@nzhost-h1 ~]$ ./fqRegister.sh --config MyConfig
--db MyDb --udtf MYFUNCTION --unregister
```

Functions and credentials unregistered successfully from database "MYDB".

This command removes the function named MYFUNCTION from the MyDb database.

---

## The fqRemote script

You use the **fqRemote.sh** script to stop, start, and manage the remote mode data connections on the NPS appliance.

### Syntax

The **fqRemote.sh** script has the following syntax.

fqRemote.sh [action]

### Inputs

The **fqRemote.sh** script takes the following inputs:

Table 6-9. The **fqRemote.sh** input options

Input	Description
start	Starts the remote mode services on the NPS host.
stop	Stops the remote mode services on the NPS host.
ping	Checks the status of the remote mode services on the NPS host to verify whether they are active and responding to requests.

Table 6-9. The **fqRemote.sh** input options (continued)

Input	Description
ps	Displays the processes that are running on the NPS host related to the remote mode functions.
connections	Displays the available connections related to the remote mode services on the NPS host.
repair	Repairs or removes any hung or no longer required connections for the remote mode functions on the NPS host.

## Description

You use the **fqRemote.sh** command to manage and troubleshoot the remote mode function connections on the NPS host. This script is a command-line interface to function calls that are available in the IBM Netezza Analytics feature. If you do not have nz user access to the NPS host, but you have database user access, you could use the function calls to perform similar tasks on the system. For more information, see “Remote service administration tasks” on page 3-24.

## The fqAdmin script

You run the **fqAdmin.sh** script to list all the UDTF functions and their configuration.

## Syntax

The **fqAdmin.sh** script has the following syntax:

```
fqAdmin.sh <command> [params]
```

## Inputs

The **fqAdmin.sh** script takes the following inputs:

Table 6-10. The **fqAdmin.sh** input options

Input	Description
help	Displays usage and syntax for the command.
list	Lists all registered connectors in all databases.
details <config>	Shows all properties for the specified Fluid Query user defined extension.
dumpConnectors <config_list> <output_path> where:<config-list> := ALL   [ <config>   [ ,<config> ] ]	Stores the configuration of connectors determined by <config-list> for future migration process in <output_path> in single tar archive. The name of tar archive is based on connector name , box name and current timestamp: <hostname>_<config>_<timestamp>.tar



## The IfqAdmin function

You can also use the IfqAdmin function to list all the UDTF functions and their configuration via SQL queries. This function is registered during IBM Fluid Query installation process.

### Usage

You can use the IfqAdmin function with one of the following options:

- IfqAdmin('help')
- IfqAdmin('list')
- IfqAdmin('details','<config>')

### Example

```
SELECT * FROM TABLE WITH FINAL  
  (IfqAdmin('details','cloudera_hive_123'));
```

**Note:** To use the IfqAdmin function registered in the SYSTEM database, use double dots as in the following example:

```
"SELECT * FROM TABLE WITH FINAL (SYSTEM..IfqAdmin('list'))"
```

---

## The FqRead function

You use the FqRead data connector function in your SQL queries to read data from service provider tables using the IBM Fluid Query data connector.

FqRead is the default name of the data connector functions, but you can specify a different name when you add them to an NPS database using the **fqRegister.sh** script. The FqRead function is an overloaded function; that is, the function has four different signatures. Each signature or form has different input arguments which provide some flexibility in the function invocations:

- FQREAD (database VARCHAR(ANY), tableName VARCHAR(ANY))

Using this form, you can specify the database name and table name from which you want to read the table data. The system runs a SELECT to obtain all the rows from the specified table. A sample call follows:

```
FqRead('mydb', 'mytable');
```

- FQREAD (database VARCHAR(ANY), tableName VARCHAR(ANY), sql VARCHAR(ANY))

In this form of the function call, you specify an SQL query that you want to run with more filtering or restrictions to limit the rows returned to NPS. In this invocation form, you cannot specify a tableName value in addition to the sql value. A sample call follows:

```
FqRead('', '', 'Select * from table_02 where code='12' limit 10000');
```

- FQREAD (database VARCHAR(ANY), tableName VARCHAR(ANY), sql VARCHAR(ANY), dataType VARCHAR(ANY))

Using this form, you can specify the database name, table name, column name, and you can override default data type in the column. This allows you to control how the data type in the column are defined on the NPS system. You can specify multiple columns separated with a comma. A sample call follows:

```
FqRead('database', 'table_01', '', 'myCol1 VARCHAR(123), otherCol FLOAT');
```

In this sample call, the myCol1 column is converted to type VARCHAR(123) in the NPS system, and the otherCol column is converted to type FLOAT.

- FQREAD(database VARCHAR(ANY), tableName VARCHAR(ANY), sql VARCHAR(ANY), targetStringSize int)

Using this form, you can specify the database name, table name, and override the default VARCHAR size. The system runs a SELECT to obtain all the rows from the table. A sample call follows:

```
FqRead('database', 'table_02', '', 400);
```

In this sample call, all of the String data types are converted to Varchar(400). Other type conversions follow the automatic type conversion mappings as described in “Data type mapping for predefined connectors” on page 3-7.

---

## The toHadoop and fromHadoop functions

You use the **toHadoop** and **fromHadoop** functions to import data from the NPS to the Hadoop cluster and to export data from the Hadoop cluster to NPS using SQL queries.

**Note:** The default behavior for the toHadoop and fromHadoop functions is to transfer the data only if target does not exist, otherwise transfer fails. This behavior is set in the XML template with the following parameter:

`nz.fq.append.mode=Create`. You can change this behavior in the configuration XML file used for a particular connection definition.

### toHadoop function

The toHadoop function is the default name for a function that sends data to the Hadoop cluster. It is equivalent to the data movement import action. You can specify a different function name when you use the **fqRegister.sh** script to register the function. The first four arguments have a specific order, but the subsequent ones provide overrides for the import configuration file fields.

- toHadoop('nps\_db', 'nps\_table');

The first two arguments are the NPS database name and the NPS table name. The function copies the data from the NPS\_DB.NPS\_TABLE to the Hadoop cluster using default Hive schema and *nps\_table* as Hive table name.

Note that if *nps\_table*, the second parameter of toHadoop, is provided in the form of `nps_schema.nps_table`, then the default Hive table name is `nps_schema_nps_table` and the target directory is `<nz.fq.output.path>/nps_schema_nps_table`.

**Important:** If the table name is enclosed in double quotation marks, you must specify two closing double quotation marks to process the name. For example:

```
MYDB.MYSCH(MYUSER)=> call toHadoop('MYDB', '"FDQM!Tb1"');
```

If you omit the second double quotations character, the function could fail with the error: `com.ibm.nz.fq.common.exception.QArchException: Input table does not exist`. Please consult the log files for more details.

- 

```
toHadoop('', 'nps_table');
```

If the NPS database name is not specified, the command uses the database in which the toHadoop function is registered.

-

```
toHadoop('nps_db', 'nps_table', 'hive_schema')
```

and

```
toHadoop('nps_db', 'nps_table', 'hive_schema', 'hive_table')
```

All of the other arguments needed to move data to the Hadoop cluster are taken from the edited `fq-remote-conf.xml` file. The file parameters can be overridden using command arguments starting at the fifth argument:

```
toHadoop('nps_db', 'nps_table', 'hive_schema', 'hive_table',  
'nz.fq.splits=12', 'nz.fq.fs.temp=/tmp');
```

If only some changes are needed in the XML configuration file, but the default NPS database, Hive schema, and Hive table should be used, you can omit those arguments from the command, for example:

```
toHadoop('', 'nps_table', '', '', 'nz.fq.splits=12', 'nz.fq.fs.temp=/tmp',  
'nz.fq.', 'nz.fq...', ...)
```

## fromHadoop function

The `fromHadoop` function is the default name for the function that downloads data from a Hadoop cluster. It is equivalent to the data movement export action. You can specify a different function name when you use the **fqRegister.sh** script to register the function. The first three arguments have a specific order, but the subsequent ones provide overrides for the export configuration file fields.

- 

```
fromHadoop('nps_db', 'nps_table');
```

First two arguments are NPS database name and NPS table name. The function copies data to the `NPS_DB.NPS_TABLE` from the Hadoop cluster using the following default HDFS path (equivalent of `nz.fq.input.path` in the data movement):

```
nz.fq.output.path + "/" + nps_table.
```

The `nz.fq.output.path` value will be taken from the edited `fq-remote-conf.xml` file.

- 

```
fromHadoop('', 'nps_table');
```

If the NPS database name is not specified, the command uses the database in which the `toHadoop` function is registered.

- Another argument is `nz.fq.input.path` which is a path on HDFS where the data is stored:

```
fromHadoop('nps_db', 'nps_table', '/nzbackup/backup1')
```

- All of the other arguments needed to copy data from the Hadoop cluster are taken from the edited `fq-remote-conf.xml` file. All of the arguments in the file can be overridden by specifying them in the command arguments, for example:

```
fromHadoop('nps_db', 'nps_table', '/nzbackup/backup1', 'nz.fq.splits=12',
'nz.fq.fs.temp=/tmp')
```

If only some changes are needed in the XML configuration file, but the default NPS database and `nz.fq.input.path` should be used, then those arguments can be omitted:

```
fromHadoop('', 'nps_table', '', 'nz.fq.splits=12', 'nz.fq.fs.temp=/tmp',
'nz.fq...', 'nz.fq...', ...)
```

## IBM support scripts

You use the support scripts to collect any valuable system information for the support team when troubleshooting problems with IBM® Fluid Query.

### The fqLogCollector script

You use the `fqLogCollector.sh` script to collect system logs for the support team when troubleshooting problems with IBM Fluid Query.

The `fqLogCollector.sh` script is located in `/nz/export/ae/products/fluidquery/fqSupportTools` on NPS, and in `/fluidqueryLocal/scripts/fqSupportTools` on Hadoop.

The script produces the output package `fqlog_<date>_<time>.tgz` in the `fqSupportTools` folder. You deliver this package to the IBM support team.

#### Syntax

The `fqLogCollector.sh` script has the following syntax.

```
fqLogCollector.sh [--hung ] [--debug ]
```

#### Inputs

The `fqLogCollector.sh` script takes the following inputs:

*Table 6-11. The fqLogCollector.sh input options*

Input	Description
<code>--hung</code>	Captures javacore for all java processes.
<code>--debug</code>	This option is for support use only.

### The fqCheckConfiguration script

You run the `fqCheckConfiguration.sh` script on NPS to verify connection from NPS to Hadoop.

The script can only be run in IBM PureData System for Analytics. It is located in `/nz/export/ae/products/fluidquery/fqSupportTools/` directory on NPS.

#### Syntax

```
fqCheckConfiguration.sh --host hostname --port port_number --provider provider_name --service service_type [options] [authentication]
```

## Inputs

Table 6-12. Input for `fqCheckConfiguration.sh` [options]

Input	Description
<code>--host</code>	Specifies the host name of the system where the service is running.
<code>--port</code>	<p>Specifies the port number for the service to which you are connecting. The command uses the following default port numbers of services:</p> <ul style="list-style-type: none"><li>• For Hive: 10000</li><li>• For SparkSql: 10000</li><li>• For Impala: 21050</li><li>• For BigSQL: 51000</li><li>• For BigSQL v1: 7052</li><li>• For DB2: 50000</li><li>• For PDOA: 50000</li><li>• For dashDB: 50000</li><li>• For Oracle: 1521</li><li>• For PDA: 5480</li></ul>
<code>--database</code>	Specifies database name for service to which you are connecting. Parameter is required for: DB2, PureData System for Operational Analytics, dashDB, IBM PureData System for Analytics, and Oracle services. For other services it is optional.
<code>--provider</code>	<p>Specifies the vendor name of the service provider. You can specify one of the following values:</p> <ul style="list-style-type: none"><li>• IBM</li><li>• Cloudera</li><li>• Horton</li><li>• Oracle</li><li>• Generic</li></ul>

Table 6-12. Input for `fqCheckConfiguration.sh` [options] (continued)

Input	Description
<code>--service</code>	Specifies the type of service to which you are connecting. You can specify one of the following values: <ul style="list-style-type: none"> <li>• Impala</li> <li>• BigSql</li> <li>• BigSqlv1</li> <li>• Hive</li> <li>• Impala</li> <li>• DB2</li> <li>• PDOA (for PureData System for Operational Analytics)</li> <li>• dashDB</li> <li>• SparkSql</li> <li>• Oracle</li> <li>• PDA (for IBM PureData System for Analytics)</li> <li>• FQDM (for the data movement support when run from NPS systems)</li> <li>• Generic</li> </ul>
<code>--config</code>	Specifies the name of the configuration file that is created after the connection is successful. If the configuration file name is <code>MyConf</code> , then the system creates a <code>MyConf.properties</code> file. The default configuration name is <code>default</code> .
<code>--driver-path</code>	Specifies the driver path to use. If not specified, default path for a given provider/service is used. Check documentation for details.
<code>--help</code>	Displays usage and syntax for the command.
<code>--debug</code>	Writes more information about the processing of the script to the log file for troubleshooting purposes.

Table 6-13. Input for the `fqCheckConfiguration.sh` [authentication]

Input	Description
<code>--username</code>	Specifies the user account name to use during authentication.
<code>--password</code>	Specifies the password for the local user name or SSL authenticated user name. You can also skip this parameter and you are automatically prompted for password.

## Usage

The following sample commands show some of the common command uses and syntax. The IP addresses and domains shown below are samples and must be replaced with the specifics in your specific service provider environment.

To check for a connection to Cloudera/Hive:

```
[nz@nzhost-h1 ~]$ ./fqCheckConfiguration.sh --host 192.0.2.1 --port 10000
--provider cloudera --service hive --username hdfs --password hdfs
```

To check for a connection to IBM/DB2:

```
[nz@nzhost-h1 ~]$ ./fqCheckConfiguration.sh --host 192.0.2.2 --port 50000 --provider IBM
--service DB2 --username db2 --password db2
```

**Note:** Only clear text user and password is supported. Other methods of authentication are not supported.

## The fqCheckJDBCconnect script

You run the fqCheckJDBCconnect.sh script on Hadoop to test the connection to IBM PureData System for Analytics or others.

The script is located in the data movement directory on Hadoop:  
/fluidqueryLocal/scripts.

### Syntax

```
fqCheckJDBCconnect.sh --host hostname --port port_number --provider provider_name --service service_type [options] [authentication]
```

### Inputs

Table 6-14. Input for fqCheckJDBCconnect.sh [options]

Input	Description
--host	Specifies the host name of the system where the service is running.
--port	Specifies the port number for the service to which you are connecting. The command uses the following default port numbers of services: <ul style="list-style-type: none"><li>• For Hive: 10000</li><li>• For SparkSql: 10000</li><li>• For Impala: 21050</li><li>• For BigSQL: 51000</li><li>• For BigSQL v1: 7052</li><li>• For DB2: 50000</li><li>• For PDOA: 50000</li><li>• For dashDB: 50000</li><li>• For Oracle: 1521</li><li>• For PDA: 5480</li></ul>
--database	Specifies database name for service to which you are connecting. Parameter is required for: DB2, PureData System for Operational Analytics, dashDB, IBM PureData System for Analytics, and Oracle services. For other services it is optional.

Table 6-14. Input for `fqCheckJDBCconnect.sh` [options] (continued)

Input	Description
<code>--provider</code>	Specifies the vendor name of the service provider. You can specify one of the following values: <ul style="list-style-type: none"> <li>• IBM</li> <li>• Cloudera</li> <li>• Horton</li> <li>• Oracle</li> <li>• Generic</li> </ul>
<code>--service</code>	Specifies the type of service to which you are connecting. You can specify one of the following values: <ul style="list-style-type: none"> <li>• Impala</li> <li>• BigSql</li> <li>• BigSqlv1</li> <li>• Hive</li> <li>• Impala</li> <li>• DB2</li> <li>• PDOA (for PureData System for Operational Analytics)</li> <li>• dashDB</li> <li>• SparkSql</li> <li>• Oracle</li> <li>• PDA (for IBM PureData System for Analytics)</li> <li>• FQDM (for the data movement support when run from NPS systems)</li> <li>• Generic</li> </ul>
<code>--help</code>	Displays usage and syntax for the command.
<code>--debug</code>	Writes more information about the processing of the script to the log file for troubleshooting purposes.

Table 6-15. Input for the `fqCheckJDBCconnect.sh` [authentication]

Input	Description
<code>--username</code>	Specifies the user account name to use during authentication.
<code>--password</code>	Specifies the password for the local user name or SSL authenticated user name. You can also skip this parameter and you are automatically prompted for password.

## Usage

The following sample commands show some of the common command uses and syntax. The IP addresses and domains shown below are samples and must be replaced with the specifics in your specific service provider environment.

To check for a connection to IBM/IBM PureData System for Analytics:



```
[nz@nzhost-h1 ~]$ ./fqCheckJDBCconnect.sh --host 192.0.2.1 --port 5480 --provider ibm  
--service pda --database system --username admin --password password
```

To check for a connection to IBM/DB2:

```
[nz@nzhost-h1 ~]$ ./fqCheckJDBCconnect.sh --host 192.0.2.2 --port 50000 --provider IBM  
--service DB2 --username db2 --password db2
```

**Note:** Only clear text user and password is supported. Other methods of authentication are not supported.



---

## Chapter 7. Appendix: Using IBM Fluid Query with MapR

Starting with version 1.7.1, you can use IBM Fluid Query to connect and retrieve data from MapR platform. You can also move data in TXT, and Hadoop specific formats between MapR and NPS.

---

### Configuring a connector to MapR

You use a generic connector functionality to connect NPS with MapR.

#### Before you begin

1. If you plan to use the connector feature, but not the data movement feature, IBM Fluid Query must be installed on NPS only.
2. Copy the following drivers from your MapR system to the NPS location `/nz/export/ae/products/fluidquery/libs/generic/MapR`:
  - `/opt/mapr/hive/hive-1.2/lib/*`
  - `/opt/mapr/hadoop/hadoop-0.20.2/lib/*`

#### About this task

The detailed process of configuring a connection and registering functions needed to query the data with generic connector is described in Chapter 3, “Data connector,” on page 3-1. This task only provides sample commands for MapR.

#### Procedure

1. Use the connection template dedicated for MapR: `connectorTemplates/MapR_hive_connection.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<connection>
  <driverClass>org.apache.hive.jdbc.HiveDriver</driverClass>
  <classPath>/nz/export/ae/products/fluidquery/libs/generic/MapR</classPath>
  <connectionURL>jdbc:hive2://9.158.143.239:10000/default;</connectionURL>
  <jdbc-properties>
    <property>
      <name>user</name>
      <value>mapr</value>
    </property>
    <property>
      <name>password</name>
      <value>mapr</value>
    </property>
  </jdbc-properties>
</connection>
```

2. Run the `fqConfigure` script to configure the connection to MapR with the following options:

```
./fqConfigure.sh --provider generic --service generic
--connectionXmlPath connectorTemplates/MapR_hive_connection.xml --config vp_mapR
```

3. Run the `fqRegister` script to register the UDTF function called, for example, `hfreadd_mapr`.

```
./fqRegister.sh --config vp_mapR --udtf hfreadd_mapr
```

## Results

You can now query the data on MapR using the registered function:

```
select * from table with final (hread_mapr('', 'nation'));
```

NATION.N_NATIONKEY	NATION.N_NAME	NATION.N_REGIONKEY	NATION.N_COMMENT
6	united kingdom	3	united kingdom
7	portugal	3	portugal
12	macau	4	aomen
8	united arab emirates	3	al imarat al arabiyah multahidah
1	canada	1	canada
10	australia	4	australia
11	japan	4	nippon
4	guyana	2	guyana
3	brazil	2	brasil
5	venezuela	2	venezuela
2	united states	1	united states of america
13	hong kong	4	xianggang
9	south africa	3	south africa
14	new zealand	4	new zealand

(14 rows)

## Running data movement between NPS and MapR

You can run data movement between NPS and the MapR platform.

**Note:** Data movement with MapR cannot be run from NPS, you must use Hadoop only.

### Before you begin

Install IBM Fluid Query on MapR. Use an additional parameter when installing to specify the mapr user:

```
./fluidquery_install.sh --hdfs_user mapr
```

### About this task

Data movement between NPS and MapR can only be run from Hadoop and it works for TXT and Hadoop-specific formats. Data movement for NZBAK format is not supported.

### Procedure

To run data movement in TXT, PARQUET, ORC, RCFILE, or AVRO format:

1. Define the configuration in the import or export XML file. Define the data format in `nz.fq.data.format` and MapR credentials in:

```
<property>
  <name>nz.fq.sql.user</name>
  <value>mapr</value>
  <description>User name. Required if User/Password authentication should be used. </descri
</property>
<property>
  <name>nz.fq.sql.password</name>
  <value>mapr</value>
  <description>Password. Required if user name was provided.</description>
</property>
```

2. Run the following command:

```
fdm.sh -conf /fluidqueryLocal/fq-export-remote-conf.xml
```



---

## Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to: This information was developed for products and services offered in the U.S.A.

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing 2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Software Interoperability Coordinator, Department 49XA  
3605 Highway 52 N  
Rochester, MN 55901  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.



Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. \_enter the year or years\_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Trademarks

IBM, the IBM logo, [ibm.com](http://www.ibm.com)<sup>®</sup> and Netezza are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (<sup>®</sup> or <sup>™</sup>), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/ or other countries.

Cloudera and the Cloudera logo are trademarks or registered trademarks of Cloudera Inc. in the USA and other countries.

Hadoop and the Hadoop elephant logo are trademarks of the Apache Software Foundation.

Hortonworks is a trademark of Hortonworks Inc. in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Red Hat is a trademark or registered trademark of Red Hat, Inc. in the United States and/or other countries.

Other company, product or service names may be trademarks or service marks of others.



---

## Accessibility features for IBM Fluid Query

Accessibility features assist users who have a disability, such as restricted mobility or limited vision, to use information technology content successfully.

### Accessibility features

IBM Fluid Query includes the following major accessibility features:

- Keyboard-only operation
- Operations that use a screen reader

IBM Fluid Query uses the latest W3C Standard, WAI-ARIA 1.0 (<http://www.w3.org/TR/wai-aria/>), to ensure compliance to US Section 508 (<http://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards>), and Web Content Accessibility Guidelines (WCAG) 2.0 (<http://www.w3.org/TR/WCAG20/>). To take advantage of accessibility features, use the latest release of your screen reader in combination with the latest web browser that is supported by this product.

The IBM Fluid Query online product documentation in IBM Knowledge Center is enabled for accessibility. The accessibility features of IBM Knowledge Center are described at [http://www.ibm.com/support/knowledgecenter/doc/kc\\_help.html#accessibility](http://www.ibm.com/support/knowledgecenter/doc/kc_help.html#accessibility).

### Keyboard navigation

This product uses standard navigation keys.

### Interface information

The IBM PureData System for Analytics user interfaces do not have content that flashes 2 - 55 times per second.

The Netezza Performance Portal web user interfaces rely on cascading style sheets to render content properly and to provide a usable experience. The application provides an equivalent way for low-vision users to use a user's system display settings, including high-contrast mode. You can control font size by using the device or web browser settings.

The Netezza Performance Portal web user interface includes WAI-ARIA navigational landmarks that you can use to quickly navigate to functional areas in the application.

### Vendor software

IBM PureData System for Analytics includes certain vendor software that is not covered under the IBM license agreement. IBM makes no representation about the accessibility features of these products. Contact the vendor for the accessibility information about its products.

## **Related accessibility information**

In addition to standard IBM help desk and support websites, IBM has established a TTY telephone service for use by deaf or hard of hearing customers to access sales and support services:

TTY service  
800-IBM-3383 (800-426-3383)  
(within North America)

## **IBM and accessibility**

For more information about the commitment that IBM has to accessibility, see IBM Accessibility ([www.ibm.com/able](http://www.ibm.com/able)).

---

# Index

## A

accessibility features for this  
product B-1

## D

data connector 5-2  
    assigning privileges to functions 3-27  
    connections, configuring 3-2  
    define service connections 6-1  
    Hadoop environments 2-2  
    IBM Netezza Analytics  
        installation 2-3  
    installation prerequisites 2-1  
    installing 2-4  
    JDBC drivers 3-2  
    local and remote mode 3-21  
    log files 5-19  
    overview 3-1  
    read function 6-13  
    register functions 6-8  
    registering functions 3-19  
    removing 2-7  
    revoking privileges from  
        functions 3-29  
    software prerequisites 2-1  
    SQL queries 3-30  
    troubleshooting configuration  
        problems 5-15  
    unregistering functions 3-32  
    upgrading 2-5  
data connector functions, registering 6-8  
data movement  
    define service connections 6-1  
    downloading Hadoop files 6-6  
    register functions 6-8  
Data movement 4-1  
    configuring 4-4  
    encrypting XML 4-27  
    export 4-14  
    import 4-5  
    installing 2-8  
    logging 5-19  
    performance 1-9  
    preinstallation 2-10  
    running 4-27  
        from NPS 4-33  
        Hadoop 4-31  
    running on other systems 4-34  
    troubleshooting 5-3  
data movement functions,  
    registering 6-8

## E

encrypting XML  
    data movement 4-27

## F

fqConfigure.sh script 6-1  
fqDownload.sh script 6-6  
FqRead function 6-13  
fqRegister.sh script 6-8  
fqRemote.sh script 6-11

## H

Hadoop connections, configuring 3-2  
Hadoop files, downloading 6-6

## I

IBM Fluid Query  
    command and function reference 6-1  
IBM Netezza Analytics, installing 2-3

## J

JDBC drivers, for data connector 3-2

## K

Kerberos configuration file 3-9  
known issues 5-2  
krb5.conf file 3-9

## L

local and remote mode functions,  
    about 3-21  
local mode  
    about 3-21  
    best practices 3-32  
    important considerations 3-21  
workload management  
    considerations 3-22

## R

remote mode  
    about 3-23  
    command for managing 6-11  
    workload management  
        considerations 3-23  
remote service  
    administration tasks 3-24  
    displaying process information 3-26  
    listing connections 3-27  
    repairing 3-27  
    starting 3-24  
    stopping 3-25  
    testing response 3-25

## S

SQL queries, running with data  
connector 3-30

## W

workload management considerations  
    local mode 3-22  
    remote mode 3-23







Part Number: 29000 Rev. 4

Printed in USA