IBM Fluid Query
Release 1.0

IBM Netezza Fluid Query User Guide

Note

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

Revised: March 25, 2015

This edition applies to IBM Fluid Query release 1.0 and to all subsequent releases until otherwise indicated in new editions.

Contents

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.

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/customercare/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 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. 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 initial offering, you can query against structured data stored within Hadoop file systems.

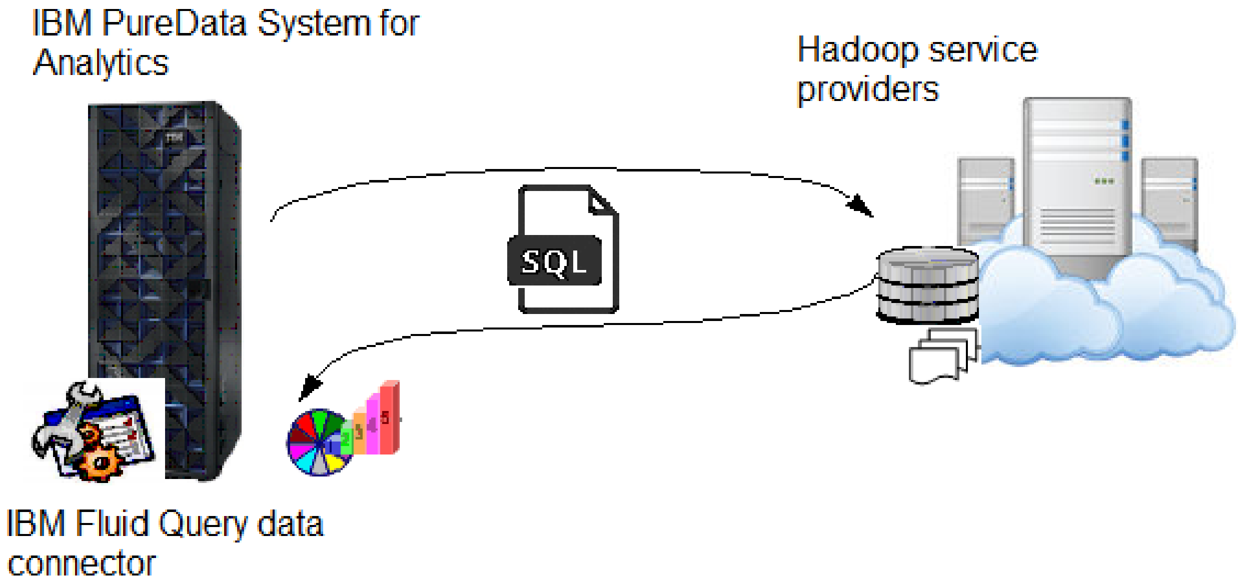
IBM PureData System for Analytics
Hadoop service providers
SQL
IBM Fluid Query data connector

Figure 1-1. IBM Fluid Query data connector environment

To use the data connector feature, you first install the data connector files on your existing IBM PureData System for Analytics appliance. You can then create connections to your Hadoop systems and service providers, such as IBM BigInsights™, Cloudera, and Hortonworks systems. You can use the data connector user-defined table functions to create SQL queries that can select from structured data stored on the Hadoop file systems, which allows you to combine the results from your Netezza Platform Software (NPS) database tables and the Hadoop data sources to create powerful combinations of analytics.

The following sections describe the system and software requirements for the data connector feature and the supported Hadoop providers. Review these sections to install the feature, and then follow the steps to configure the connections, register the data connector functions, and use the functions in your SQL queries. There are also sections for important workload management considerations when running these data connector queries on your system.

Data connector installation overview

Software prerequisites

To use the data connection feature, you must have an NPS system, a Hadoop environment, 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.

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

Table 1-1. 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.x	3.0.2	3.2

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 1-2. List of supported services and Hadoop environments

Service	Provider		
	IBM BigInsights versions 2.1, 3.0	Cloudera versions 4.7, 5.3	Hortonworks version 2.1, 2.2
Hive2	Yes	Yes	Yes
BigSql	Yes	No	No
Impala	No	Yes	No

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

JDBC driver prerequisites

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

The following table lists the supported Hadoop service providers, the location on the NPS appliance where the JDBC drivers must be stored, and the required JDBC driver files. 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 1-3. List of JDBC drivers for the supported Hadoop service providers

Provider	Service	NPS path	Sample file list
IBM	Hive2	`./libs/ibm/hive/`	commons-configuration-1.6.jar commons-logging-1.1.1.jar hadoop-core-2.2.0-mr1.jar hive-exec-0.12.0.jar hive-jdbc-0.12.0.jar hive-metastore-0.12.0.jar hive-service-0.12.0.jar hive-shims-0.12.0.jar httpclient-4.2.5.jar httpcore-4.2.4.jar libfb303-0.9.0.jar libthrift-0.9.0.jar log4j-1.2.17.jar slf4j-api-1.6.1.jar slf4j-log4j12-1.6.1.jar
	BigSql	`./libs/ibm/bigsql/`	db2jcc.jar
	BigSqlv1	`./libs/ibm/bigsqlv1/`	bigsql-jdbc-driver.jar
Cloudera	Impala/ Hive2	`./libs/cloudera/hive/`	commons-collections.jar commons-configuration.jar commons-lang.jar commons-logging-1.0.4.jar guava.jar hadoop-annotations-2.0.0-cdh4.6.0.jar hadoop-auth-2.0.0-cdh4.6.0.jar hadoop-common-2.0.0-cdh4.6.0.jar hadoop-mapreduce-client-core.jar hive-exec-0.10.0-cdh4.6.0.jar hive-jdbc-0.10.0-cdh4.6.0.jar hive-metastore-0.10.0-cdh4.6.0.jar hive-service-0.10.0-cdh4.6.0.jar hive-shims-0.10.0-cdh4.6.0.jar httpclient-4.2.5.jar httpcore-4.2.5.jar libfb303-0.9.0.jar log4j-1.2.16.jar slf4j-api-1.6.4.jar slf4j-log4j12-1.6.1.jar
Hortonworks	Hive2	`./libs/horton/hive/`	commons-codec-1.4.jar commons-collections-3.2.1.jar commons-configuration-1.6.jar commons-logging-1.1.3.jar hadoop-auth-2.4.0.2.1.7.0-784.jar hadoop-common-2.4.0.2.1.7.0-784.jar hadoop-mapreduce-client-core-2.4.0.2.1.7.0-784.jar hive-common-0.13.0.2.1.7.0-784.jar hive-exec-0.13.0.2.1.7.0-784.jar hive-jdbc-0.13.0.2.1.7.0-784.jar hive-service-0.13.0.2.1.7.0-784.jar httpclient-4.2.5.jar httpcore-4.2.5.jar libthrift-0.9.0.jar log4j-1.2.17.jar slf4j-api-1.7.5.jar

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 1-5 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 JDBC drivers

If you use the IBM BigInsights BigSql and Hive2 services, you could 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 the table above.

Obtaining Cloudera JDBC drivers

If you use the Cloudera Impala and Hive2 services, you could obtain the JDBC driver files as follows:

1. Copy the required JAR files (shown in table above) 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 only need the files shown in the table above.
3. Save the JDBC files to the directory shown in the table above.

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

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 above) 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 only need the files shown in the table above.
3. Save the JDBC files to the directory shown in the table above.

Troubleshooting missing JDBC drivers

If you encounter any problems finding or obtaining the JDBC files listed in the table above, 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.

Installing IBM Netezza Analytics

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

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.

1. Obtain the IBM Netezza Analytics software from IBM Fix Central at http://www.ibm.com/support/fixcentral.
2. 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
```

Installing the data connector

Follow these steps to install the data connector software on the IBM PureData System for Analytics appliance.

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 v1.0. 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 tar file using the following command:
   ```
   gunzip nz-fluidquery-version.tar.gz
   ```
5. Decompress the resulting tar file using the following command:
   ```
   tar -xvf nz-fluidquery-version.tar
   ```

   Sample output follows:
   ```
   ./
   ./fluid-query-sql-v1.0.tar
   ./fluid-query-import-export-v1.0.tar
   ```
6. Decompress the `fluid-query-sql-v1.0.tar` file using a command similar to the following:
   ```
   tar -xvf fluid-query-sql-v1.0.tar
   ```

   Sample output follows:
   ```
   java/
   perl/
   perl/fluidquery/
   perl/fluidquery/framework/
   perl/fluidquery/plugin/
   fluidquery_install.pl
   installPack.tar
   java/ibm-java-sdk-6.0-16.0-linux-i386.tgz
   perl/fluidquery/framework/fluidquery_install_prerun.pm
   perl/fluidquery/framework/fluidquery_install_run.pm
   perl/fluidquery/framework/fluidquery_install_setup.pm
   perl/fluidquery/framework/fluidquery_util.pm
   ```
7. Run the installation command:
   ```
   ./fluidquery_install.pl
   ```

   Sample output follows:

```
--------------------------------------------------------------------------------
IBM Fluid Query Installer
(C) Copyright IBM Corp. 2015 All rights reserved.
--------------------------------------------------------------------------------
Logging to file /nz/var/log/fluidquery_install/fluidquery_install201525105535.log

Checking for previous installation of IBM Fluid Query ... [OK]
Checking existing Java for IBM Netezza Analytics ... [OK]
Checking for existing installation of IBM Netezza Analytics ... [OK]

Beginning installation of IBM Fluid Query
Extracting installPack.tar to /nz/export/ae/products/fluidquery ... [OK]
Updating script files for execute permission ... [OK]
Existing Java found which needs to be updated
Backing up old Java to /nz/export/ae/languages/java/java_sdk/ibm-java-i386-60 to
/nz/export/ae/languages/java/java_sdk/ibm-java-i386-60.orig ... [OK]
Beginning installation process for IBM Netezza Analytics-compatible Java
Extracting Java to /nz/export/ae/products/fluidquery ... [OK]
Linking Java from /nz/export/ae/products/fluidquery to
/nz/export/ae/languages/java/java_sdk/ ...
/nz/export/ae/languages/java/java_sdk/ibm-java-i386-60 already exists, unlinking
and relinking ...

Finished installing IBM Fluid Query software

Refer to the IBM Fluid Query User Guide for instructions to configure and use
the feature.

Ending installation, for details see log file at
/nz/var/log/fluidquery_install/fluidquery_install201525105535.log
```

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

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 1-5 for more information. After you install IBM Netezza Analytics, run the **fluidquery_install.pl** script again.

If you want to upgrade the data connector software, see "Upgrading the data connector" for specific instructions.

Upgrading the data connector

If there is a new kit or fix pack available for the data connector software, follow these steps to upgrade the software on the IBM PureData System for Analytics appliance.

The data connector upgrade procedure uses the same **fluidquery_install.pl** script as the installation process. If the **fluidquery_install.pl** detects that the data connector software 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.

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 v1.0. 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 tar file using the following command:

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

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

   Sample output follows:

   ```
   ./
   ./fluid-query-sql-v1.0.tar
   ./fluid-query-import-export-v1.0.tar
   ```
6. Decompress the `fluid-query-sql-v1.0.tar` file using a command similar to the following:

   ```
   tar -xvf fluid-query-sql-v1.0.tar
   ```

   Sample output follows:

   ```
   java/
   perl/
   perl/fluidquery/
   perl/fluidquery/framework/
   perl/fluidquery/plugin/
   fluidquery_install.pl
   installPack.tar
   java/ibm-java-sdk-6.0-16.0-linux-i386.tgz
   perl/fluidquery/framework/fluidquery_install_prerun.pm
   perl/fluidquery/framework/fluidquery_install_run.pm
   perl/fluidquery/framework/fluidquery_install_setup.pm
   perl/fluidquery/framework/fluidquery_util.pm
   ```
7. Run the installation command:

   ```
   ./fluidquery_install.pl
   ```

   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.
   -------------------------------------------------------------------------------
   Logging to file /nz/var/log/fluidquery_install/fluidquery_install201535141443.log

   Checking for previous installation of IBM Fluid Query ... [OK]
   A previous installation of IBM Fluid Query was located.
   Checking existing Java for IBM Netezza Analytics ... [OK]
   Checking for existing installation of IBM Netezza Analytics ... [OK]
   Would you like to upgrade the existing IBM Fluid Query installation? (y/n) [y]Enter
   Backing up IBM Fluid Query data connector

   Backing up existing Fluid Query installation to /nz/export/ae/products/fluidquery.
   backup.201535141445 ... [OK]

   Beginning installation of IBM Fluid Query
   Extracting installPack.tar to /nz/export/ae/products/fluidquery ... [OK]
   Updating script files for execute permission ... [OK]
   Existing Java found which needs to be updated
   Backing up old Java to /nz/export/ae/languages/java/java_sdk/ibm-java-i386-60 to
   /nz/export/ae/languages/java/java_sdk/ibm-java-i386-60.orig ... [OK]
   Beginning installation process for IBM Netezza Analytics-compatible Java
   Extracting Java to /nz/export/ae/products/fluidquery ... [OK]

   Finished installing IBM Fluid Query software
   ```

```
Refer to the IBM Fluid Query User Guide for instructions to configure and use
the feature.

Ending upgrade of IBM Fluid Query data connector utility
Successfully completed
For details see log file at /nz/var/log/fluidquery_install/
fluidquery_install201535141443.log
```

After a successful installation, the installer automatically places the latest data connector 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.

The data connector removal is a manual process. 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.

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. If you use the Netezza Analytics application for reasons other than the IBM Fluid Query feature, you must restore the Java libraries link to ensure that your Netezza Analytics applications continue to run.
   a. Change to the directory where Java is installed:
      ```
      cd /nz/export/ae/languages/java/java_sdk/
      ```
   b. Remove the link to the data connector Java location `/nz/export/ae/products/fluidquery/ibm-java-i386-60`:
      ```
      rm ibm-java-i386-60
      ```
   c. Restore the Java libraries in the `/nz/export/ae/languages/java/java_sdk/ibm-java-i386-60.orig` location:
      ```
      cp -Hr /nz/export/ae/languages/java/java_sdk/ibm-java-i386-60.orig
      /nz/export/ae/languages/java/java_sdk/ibm-java-i386-60
      ```
   d. Remove the backup Java libraries:
      ```
      rm -rf /nz/export/ae/languages/java/java_sdk/ibm-java-i386-60.orig
      ```
5. When you are ready to delete the contents of the `fluidquery` directory, use the **`cd ..`** command to change to the `/nz/export/ae/products` parent directory.
6. Use the **`ls`** command to review the number of `fluidquery` directories, including backup copies, that are present in the directory.

7. Type the following command to remove the fluidquery directory:

   ```
   rm -rf fluidquery
   ```
8. If there are other directories, such as fluidquery.backup.*date_id* directories that you want to remove as well, use a command similar to the following to remove each of those backup directories, where *date_id* is the unique suffix for each backup directory.

   ```
   rm -rf fluidquery.backup.date_id
   ```

After a successful removal, the /nz/export/ae/products directory should not have any fluidquery subdirectories unless you chose to leave a specific backup or archive. The databases should no longer have any form of the FqRead functions registered for use with user queries.

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.

Configuring a connection

To use the data connector to query data on Hadoop environments, you must define a connection to each Hadoop service that you plan to query and use.

You must have installed the data connector software and the JDBC drivers for your Hadoop service providers before you can configure a connection.

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
     ```

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

If the command fails and displays an error message, you can try re-running command and adding the `--debug` option for more troubleshooting information. The failed command still creates a configuration properties file, but do not use that configuration file during function registrations because it could contain incorrect or incomplete information. For more information about the script and its options, see "The fqConfigure script" on page 1-26.

Kerberos configuration file

If your Hadoop 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 pathname to the file using the `--krb5-conf` argument of the **`fqConfigure.sh`** script. Also make sure that you save the file to the same pathname on both NPS hosts (HA1 and HA2) so that the file is available if an NPS host failover occurs.

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 driver prerequisites" on page 1-2 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.

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

Registering the data connector functions

To call and use the data connector functions in your queries, you must register the data connector functions in one or more NPS databases.

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

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):
     ```
     ./fqRegister.sh
     Functions and credentials are successfully registered in database "MAINDB".
     ```
   - To register the functions using the name HdpRead() in the database named MyDb:
     ```
     ./fqRegister.sh --udtf HdpRead --db MyDb
     Functions and credentials are successfully registered in database "MYDB".
     ```

The registration function adds the data connector functions to the specified database. 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 1-13.

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 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 1-20 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 1-29.

For more information about the FqRead() function and the four supported input forms, see "The FqRead function" on page 1-32.

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.

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.

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 creating a database (if it does not exist), creating 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.

6. For users who will create and run queries that include the data connector functions, you must also grant them Execute privilege to the functions.

   a. Connect to the database where the data connector functions have been registered as the database admin user, database owner, or as a member of the *db*_inzaadmins group.

   b. Grant the database user privileges to execute the data connector functions:

      ```
      grant execute on function_name (varchar(any),varchar(any))
           to user_name;
      grant execute on function_name (varchar(any),varchar(any),varchar(any))
           to user_name;
      grant execute on function_name (varchar(any),varchar(any),varchar(any),
           varchar(any)) to user_name;
      grant execute on function_name (varchar(any),varchar(any),varchar(any),
           integer) to user_name;
      ```

      As a tip, you can grant Execute privileges to a group such as the *db*_inzausers group rather than a specific user, and all of the members of the group will have Execute access to the data connector functions. You can then manage the privileges by adding users to or removing users from the group.

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.

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 1-13.

You can also selectively revoke privileges for a specific user.

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.
     a.
     b. Revoke the database user privileges for executing the data connector functions:

        ```
        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;
        ```

        As a 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.

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 1-13.

Unregistering the data connector functions

You can unregister the data connector functions to remove them from your NPS database.

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

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 option to unregister or remove the functions. Sample commands follow.
   - To unregister the functions named HdpRead() in the database named MyDb:

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

   - If you specify only the --unregister option, the command uses the /nz/export/ae/products/fluidquery/default.properties file to search for the function in udtfname field and removes it from the database defined by NZ_DATABASE.

     ```
     ./fqRegister.sh --unregister
     Functions and credentials unregistered successfully from database "MAINDB".
     ```

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 1-29.

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.

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 1-13. 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.

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 ( hpdread('', '', 'select
   ```

```
   sum(col1) from bidb.tab1'));
     1
   ----
    10
```

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 (fqread('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 (fqread('mydb','employees'));
 EMPLOYEES.EMP_ID | EMPLOYEES.NAME | EMPLOYEES.SALARY | EMPLOYEES.ADDRESS
------------------+----------------+------------------+-------------------
                1 | Luis           |               10 | xyz
```

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 (fqread('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 (fqread('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 (fqRead('', '','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 (fqRead('', '','select parts.c1,
orders.c1 as c2 from parts join orders on parts.c1 = orders.c1'));
 C1 | C2
----+----
 23 | 23
```

Use 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 1-16 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;
```

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.

With local mode, users can run concurrent queries to different Hadoop service providers if they register functions specific to each service provider.

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 MINUMUM 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 FQRule 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 only one remote mode service at a time, and thus can support remote mode queries to only one Hadoop service provider at a time. If your users typically query several different Hadoop services with concurrent queries, consider remote mode for the service that has the highest usage. Connections to other Hadoop services must use local mode, or you must stop the remote mode service and start a new one to a different Hadoop provider.

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.

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 is 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));
   ```

Data connector known issues

Note the following important behaviors and limitations for this release of 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 the **fqConfigure.sh** script fails, it creates a configuration properties file, but do not use that configuration file during function registrations. The functions will fail to run and could result in hangs due to incomplete or erroneous configuration information.
- 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 determine 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 “Use views to simplify user queries” on page 1-17, 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.

Data connector command reference

This section contains the descriptions of the scripts and commands used with the data connector feature.

The fqConfigure script

You use the **fqConfigure.sh** script to configure connections to the Hadoop service providers.

Syntax

The **fqConfigure.sh** script has the following syntax.

```
fqConfigure.sh --host hostname [--port number ]
--provider name --service service_name [--help] [--version]
[authentication options]
```

Inputs

The **fqConfigure.sh** script takes the following inputs:

Table 1-4. The **fqConfigure.sh** *input options*

Input	Description
--help	Displays usage and syntax for the command.
--version	Displays the version and build information for the data connector utilities.
--host	Specifies the host name of the system where the Hadoop service is running.
--port	Specifies the port number for the Hadoop service to which you are connecting. The command uses the following default port numbers for the Hadoop services: • For Hive, 10000 • For Impala, 21050 • For BigSQL, 51000 • For BigSQL v1, 7052 If your Hadoop administrator started the Hadoop services on custom ports, use the --port option to specify the port for your service environment.
--provider	Specifies the vendor name of the Hadoop provider. You can specify one of the following values: • IBM • Cloudera • Hortonworks

Table 1-4. The **fqConfigure.sh** input options (continued)

Input	Description
`--service` *service_name*	Specifies the type of Hadoop service to which you are connecting. You can specify one of the following values: • Impala • BigSql • BigSqlv1 • Hive
`--config` *file_name*	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 `MyConf.properties` 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.
`--varchar-size` *bytes*	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.)
`--debug`	Writes more information about the processing of the script to the log file for troubleshooting purposes.

Authentication options

For each Hadoop connection, you can specify SSL authentication, local user account and password authentication, or Kerberos authentication. You cannot specify both the local and Kerberos options, or both the SSL and Kerberos options.

Table 1-5. 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` *path name*	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` *password*	Specifies the custom truststore for authentication. If you specify a custom truststore, you must also provide a password.

Table 1-6. User and password authentication options

Input	Description
`--username` *name*	Specifies the user account name to use during authentication.
`--password` *password*	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.

Table 1-7. Kerberos authentication options

Input	Description
`--service-principal` *name*	Specifies the target service principal. This value must be the same user principal that was used when starting the service. (For example: impala/ myhost.example.com@example.com)
`--client-principal` *name*	Specifies the Kerberos client principal.
`--krb5-conf` *path name*	Specifies the full path to the Kerberos configuration file, as described in "Kerberos configuration file" on page 1-11. If you do not specify a `--krb5-conf` option, the command uses the default value `/etc/krb5.conf`.
`--password` *password*	Specifies the password for the client-principal. The `--password` and `--keytab` options have the following behaviors: • If you specify `--keytab` but not `--password`, the command uses the specified keytab file. • If you specify `--keytab` and `--password`, the command creates a new keytab in the specified keytab path. • If you specify `--password` but not `--keytab`, the command creates the default keytab. • If you do not specify `--password` or `--keytab`, the command prompts you for a password.
`--keytab` *path name*	Specifies the full path to the keytab that you are using. For more information about the `--keytab` and `--password` processing, see the `--password` description.

Description

You use the **`fqConfigure.sh`** command to create a connection to a Hadoop 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.

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 Hadoop 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: 1.0.150303-128:
```

The fqRegister script

You use the **fqRegister.sh** script to register the connector functions in a target Netezza database.

Syntax

The **fqRegister.sh** script has the following syntax.

```
fqRegister.sh [--help] [--user username] [--pw password]
[--db name] [--udtf name]
[--config name] [--remote] [--xmx] [--xms]
[--unregister] [--debug]
```

Inputs

The **fqRegister.sh** script takes the following inputs:

Table 1-8. The **fqRegister.sh** *input options*

Input	Description
`--help`	Displays usage and syntax for the command.
`--user` *name*	Specifies the database user account to access the Netezza database and register the functions. The default is the value of the NZ_USER variable.
`--pw` *password*	Specifies the password for the database user account. The default is the value of the NZ_PASSWORD variable.
`--db` *name*	Specifies the Netezza database name where function is registered. The default is the value of the NZ_DATABASE variable.

Table 1-8. The **fqRegister.sh** *input options (continued)*

Input	Description
`--udtf` *name*	Specifies the name of the user-defined table function that you are registering. The default name is FqRead for JDBC-related services.
`--config` *name*	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 **fqConfigure.sh** script. When specifying a value, do not include the .properties suffix of the filename. The default configuration name is `default`.
`--remote`	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 Local/Remote mode.
`--xmx` *value*	Specifies the Java xmx property used by Netezza Analytics when running the registered function. The default value is -Xmx1024M.
`--xms` *value*	Specifies the Java xms property used by Netezza Analytics when running registered function. The default value is -Xms256M.
`--unregister`	Specifies that you want to unregister, or remove, the function from the specified database.
`--debug`	Writes more information about the processing of the script to the log file for troubleshooting purposes. 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. 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.

Description

You use the **fqRegister.sh** command to add or register the Hadoop-related functions for use in a Netezza database. 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 databas "MYDB" failed` 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 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 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 1-9. 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.

Table 1-9. The **fqRemote.sh** *input options (continued)*

Input	Description
`ping`	Checks the status of the remote mode services on the NPS host to verify whether they are active and responding to requests.
`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 1-22.

The FqRead function

You use the FqRead function in your SQL queries to read data from Hadoop 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 Hadoop 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 a 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), targetSchema VARCHAR(ANY))

  Using this form, you can specify the database name, table name, and you can override a default column name and data type. This allows you to control how the column name and its data type are defined on the NPS system. (In this context, the schema is the data definition, not a schema object within a database.) A sample call follows:

```
FqRead('database', 'table_01', '', 'myCol1 VARCHAR(123), otherCol FLOAT');
```

In this sample call, the myCol1 VARCHAR(123) column is saved in the NPS system as a column named otherCol of 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 conversions.”

Data type conversions

When reading data from the Hadoop service providers, the NPS database uses the following data type conversion rules.

The following Hadoop data types are converted into the following IBM Netezza Analytics data types:

- BOOLEAN => NzaeDataTypes.NZUDSUDX_BOOL;
- REAL => NzaeDataTypes.NZUDSUDX_DOUBLE;
- DOUBLE => NzaeDataTypes.NZUDSUDX_DOUBLE;
- FLOAT => NzaeDataTypes.NZUDSUDX_DOUBLE;
- DECIMAL => NzaeDataTypes.NZUDSUDX_DOUBLE;
- INT => NzaeDataTypes.NZUDSUDX_INT32;
- INTEGER => NzaeDataTypes.NZUDSUDX_INT32;
- SMALLINT => NzaeDataTypes.NZUDSUDX_INT16;
- STRING => NzaeDataTypes.NZUDSUDX_VARIABLE (1000);
- TIMESTAMP => NzaeDataTypes.NZUDSUDX_TIMESTAMP;
- DATE => NzaeDataTypes.NZUDSUDX_DATE;
- TINYINT => NzaeDataTypes.NZUDSUDX_INT8;
- BIGINT => NzaeDataTypes.NZUDSUDX_INT64;
- BINARY => NzaeDataTypes.NZUDSUDX_VARBINARY;
- CHAR => NzaeDataTypes.NZUDSUDX_VARIABLE (1000);
- CHARACTER => NzaeDataTypes.NZUDSUDX_VARIABLE (1000);
- CLOB => NzaeDataTypes.NZUDSUDX_VARIABLE (1000);
- VARCHAR => NzaeDataTypes.NZUDSUDX_VARIABLE (1000);

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

Chapter 2. 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.

Export Data transfer from Hadoop to NPS.

The transfer can occur in three modes. You can set these modes in the **`Compression properties`** section of the XML configuration files.

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: Note that you can store data in compressed mode on Hadoop, but only for backup/restore purposes. Tables stored in the compressed format on Hadoop cannot be queried and modified.

The data movement feature is supported on systems delivered by BigInsights, Hortonworks, and Cloudera. The supported infrastructures are Hive 1 and Hive 2.

Data type conversion

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 2-1.

Data type on NPS	Data type on Hadoop
boolean	Boolean
real, double	DOUBLE, FLOAT
byteint, smallint, integer, bigint	TINYINT, SMALLINT, INT, BIGINT, BIGINT
numeric	STRING
Char, VCHAR, NCHAR, NVARCHAR	STRING
date	STRING
timestamp	STRING
time	STRING
interval	STRING

Uninstalling the product

To uninstall the product, you must delete all the IBM Fluid Query files from the local file system and from HDFS.

When attempting to delete the folders, you may receive an error that they are not empty. If it occurs, create backup copies of the required files and manually delete the folder contents one by one. Then remove the parent directories.

1. As the user that created the installation folder, remove the files from the local file system:
   - If you used the default location during the installation, run the following command:

```
rm /fluidqueryLocal/nzcodec.jar /fluidqueryLocal/nzjdbc3.jar /fluidqueryLocal/nzetc
rmdir /fluidqueryLocal
```

     If you had created the directory as the root user, you can only remove it as the root user.
   - If you set the **-hadoop_path <local_hadoop_path>** parameter when running the installation script, run the following command:

```
rm <local_hadoop_path>/nzcodec.jar <local_hadoop_path>/nzjdbc3.jar <local_hadoop_path>/nzetc
rmdir <hadoop_home_path>
```

     Note: This is the location where nzcodec.jar, nzjdbc3.jar, and nzetc files are placed on the local file system.
2. Run the following command to remove the files from HDFS:

```
hadoop fs -rm /fluidquery/nzcodec.jar /fluidquery/nzjdbc3.jar /fluidquery/nzetc
hadoop fs -rmdir /fluidquery
```

   Note: For older Hadoop systems, for example BigInsights 2.x, use the following command: **hadoop fs -rmr /fluidquery**

Installing and upgrading the data movement feature

Before you can use the data movement functionality, you must first download the IBM Fluid Query package and install a set of files on your Hadoop nodes.

The fluidquery_install.pl installation script copies files to the following folders in the root (/) directory:

- /fluidqueryLocal on the local file system.
- /fluidquery on HDFS.

Before you run the script, you must log in as root and create the /fluidqueryLocal folder on the local file system. Then you must change its owner and group to the user that will be installing and using the data movement feature:

```
chown <localuser>:<localuser> /fluidqueryLocal
```

After that, you can run the fluidquery_install.pl installation script as <*localuser*>.

Note: Attempting to run the script as root will fail, as the root user cannot create folders on HDFS.

The installation and upgrade procedures of IBM Fluid Query are the same.

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

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. Extract both packages in the same location:

   ```
   tar -xvf fluid-query-sql-v1.0.tar
   tar -xvf fluid-query-import-export-v1.0.tar
   ```

6. Run the installation script with the **`-datamove`** parameter:

   ```
   ./fluidquery_install.pl -datamove
   ```

Configuring the data movement feature

Choose the configuration type based on your Hadoop platform.

All configuration for the data movement feature is performed on the Hadoop side. You do not need to perform any steps on NPS.

Data import configuration

You need to customize the XML configuration file before you begin importing data from NPS to Hadoop.

Before you can run the import command, make sure that you edit and provide all the required properties in the `fq-import-conf.xml` configuration file. The file can be found in the `fluid-query-import-export-<`*`version`*`>.tar` package that you extracted as part of the installation process. The default values are already set in the file.

General configuration

-

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

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

-
  ```
  <property>
   <name>nz.fq.clean.before.import</name>
   <value>true</value>
  </property>
  ```

  The **nz.fq.clean.before.import** property defines whether all files that start with part-0000 are to be removed from the target folder that is specified in property **nz.fq.output.path** before running the import. It prevents the conflict that occurs when performing multiple imports to the same directory. It does not affect directories, only files.

-
  ```
  <property>
   <name>nz.fq.sql.metadata</name>
   <value>true</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**. If you set it to **false**, the table will not be created in Hive and IBM Fluid Query will only import data files and put them on HDFS.

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

  The **nz.fq.hive.schema** property sets the target schema name in Hive under which all imported tables are created. If it does not exist, it is automatically created. If the property is not set, the default schema is used.

HDFS properties

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

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

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

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

-

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

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

Compression properties

-

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

  The **nz.fq.compress** property defines whether to transfer NPS data in compressed internal format.

-

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

  The **nz.fq.output.compressed** property defines whether the transferred data is stored on Hadoop in compressed internal format. Depends on the **nz.fq.compress** setting.

NPS properties

-

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

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

-

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

  The **nz.fq.tables** property provides a comma-separated list of NPS tables. Include double quotations around delimited table names. The format of this value is **<SCHEMA>.<table>**.

  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.

-

```
<property>
 <name>nz.fq.nps.server</name>
 <value>hostname.ibm.com</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.port</name>
   <value>5480</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.user</name>
   <value>admin</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.password</name>
   <value>password</value>
  </property>
  ```

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

-

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

  The **nz.fq.nps.where** property specifies the SQL WHERE clause that is used for selecting the data to transfer.

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

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

After customizing the `fq-import-conf.xml` file, you can proceed to “Performing the data movement” on page 2-12.

Data export configuration

You need to customize the XML configuration file before you begin exporting data from Hadoop to NPS.

Before you can run the export command, make sure that you edit and provide all the required properties in the `fq-export-conf.xml` configuration file. The file can be found in the `fluid-query-import-export-<`*`version`*`>.tar` package that you extracted as part of the installation process. The default properties are already set in the file.

General configuration

-
  ```
  <property>
   <name>nz.fq.command</name>
   <value>export</value>
  </property>
  ```

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

HDFS properties

-
  ```
  <property>
   <name>nz.fq.input.path</name>
   <value>/nzbackup/fqtest1</value>
  </property>
  ```

  The **nz.fq.input.path** parameter sets the directory on HDFS where the retrieved data is stored.

-

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

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

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

NPS properties

-

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

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

-

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

  The **nz.fq.table** property provides a comma-separated list of NPS tables. Include double quotations around delimited table names. The format of this value is **<SCHEMA>.<table>**.

  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.

-

  ```
  <property>
   <name>nz.fq.nps.server</name>
   <value>hostname.ibm.com</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.port</name>
   <value>5480</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.user</name>
   <value>admin</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.password</name>
   <value>password</value>
  </property>
  ```

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

-
  ```
  <property>
   <name>nz.fq.nps.ssl</name>
   <value>false</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.cacertificate</name>
   <value></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.exttab.columns</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.schema</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. The default value is empty string - the schema of the target table is used then. Detailed syntax is described in "Transient External Tables" in *IBM Netezza Data Loading Guide*.

After customizing the fq-export-conf.xml file, you can proceed to “Performing the data movement” on page 2-12.

Exporting previously imported 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, 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. Parameters **nz.fq.exttab.columns** and **nz.fq.exttab.schema** are not needed in such case and are ignored.
- Use the HDFS directory where data files are located, for example:

  ```
  <import_set_path>/<table_name>/table
  ```

  In this case, parameters **nz.fq.exttab.columns** and **nz.fq.exttab.schema** are required and the table must exist in NPS.

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. FluidQuery 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 then 30. Usually, the best performance can be achieved with 24 streams. The number of streams can be set in the XML configuration files (**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.

FluidQuery 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. The former setting (**`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 FluidQuery 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 then the speed of a drive. Problems appear when there are not enough drives and more then one process is written to the same disk.

Performing the data movement

After you have configured the XML configuration file, you can use the IBM Fluid Query tool to transfer data between NPS and Hadoop.

Depending on what kind of transfer you want to run, configure one of the XML configuration files.

1. If you want to run an import operation using local connection to Hive (local mode), you must first export the classpath for the Hadoop job:

   For BigInsights:
   Run the following command:

```
export HADOOP_CLASSPATH="/opt/ibm/biginsights/hive/lib/*:/opt/ibm/biginsights/hive/conf/:/fluidqueryLocal/nzjdbc3.jar"
```

   For Hortonworks:
   Run the following command:

```
export HADOOP_CLASSPATH="/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/share/cmf/cloudera-navigator-server/libs/cdh4/*:/fluidqueryLocal/nzjdbc3.jar"
```

   For Cloudera 5:
   Run the following command:

```
export HADOOP_CLASSPATH="/etc/hive/conf/:/usr/share/cmf/cloudera-navigator-server/libs/cdh5/*:/fluidqueryLocal/nzjdbc3.jar
```

   For Cloudera QuickStart:
   Run the following command:

```
export HADOOP_CLASSPATH="/etc/hive/conf/:/usr/lib/hive/lib/*:/fluidqueryLocal/*"
```

2. From the command line, run `nzcodec.jar` and provide the XML configuration file as the parameter. For example, to perform a data import, run the following command:

```
hadoop jar /fluidqueryLocal/nzcodec.jar -conf fq-import-conf.xml -libjars /fluidqueryLocal/nzcodec.jar,/fluidqueryLocal/nzjdbc3.jar
```

   Note: To perform data export, run the same command changing only the **`-conf`** parameter to point to a different XML configuration file.

Data transfer is performed based on the properties that you have set in the configuration file.

Logging

For troubleshooting purposes, you can configure logging for the data movement feature.

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
```

Logs will be produced in a `fq.log` file in the same directory.

Known issues and limitations

Refer to this topic for information on the limitations of IBM Fluid Query.

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 lost

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 a Hadoop table with \N null values fails

By default, tables that are created in the Hadoop service use `\N` as the null value. However, if you export a table with `\N` nulls, the export fails because NPS cannot parse that value.

To export a table created in Hadoop, the table properties must set the null value to `NULL` instead of `\N`. For example:

```
CREATE TABLE MyTbl ... TBLPROPERTIES("serialization.null.format"="NULL")...
```

If you have an existing Hadoop table that uses the default \N null character, you could create a new table that uses NULL as the null value and then export the new table. For example, you could use a command such as `CREATE TABLE MyTbl TBLPROPERTIES("serialization.null.format"="NULL") AS SELECT FROM OldTable...` to create a new table that you could export to NPS.

Data movement troubleshooting

Refer to the following section in case if you are experiencing problems with the data movement feature.

Error: `java.lang.ClassNotFoundException: org.netezza.Driver`

Solution: Add the following parameter to the **`hadoop jar`** command:

```
-libjars nzjdbc3.jar,nzcodec.jar,nzetc
```

Errors: `Exception: hive-site.xml cannot be found in classpath` and `java.lang.NoClassDefFoundError: org/apache/hadoop/hive/ql/...`

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: `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;
```

Tables imported to BigInsights are not visible for BigSQL3

Solution: In order to see in BigSQLv3 the tables that were created in Hive, you need to call a sync objects procedure. Run the following command:

```
CALL SYSHADOOP.HCAT_SYNC_OBJECTS('schema_name', '.*', 'a', 'REPLACE', 'CONTINUE');
```

, where **schema_name** is the name of your database on Hive.

Warning: `WARN util.GenericOptionsParser: options parsing failed: Missing argument for option: libjars`

This warning message appears when you do not provide any value for the **`-libjars`** parameter when running a data transfer. It also produces additional error messages that should be ignored in this context:

```
[main] ERROR com.ibm.nz.fq.FqConfiguration  - Could not load data from configuration.
Please verify that your .xml name is correct
ERROR fq.FqConfiguration: Could not load data from configuration. Please verify that your .xml name is correct
Exception in thread "main" java.lang.Exception: Could not load data from configuration.
Please verify that your .xml name is correct
    at com.ibm.nz.fq.FqConfiguration.initialize(FqConfiguration.java:61)
    at com.ibm.nz.fq.NzTransfer.run(NzTransfer.java:96)
    at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
    at com.ibm.nz.fq.NzTransfer.main(NzTransfer.java:88)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:606)
    at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
    at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
```

To avoid these errors, make sure to provide correct values with the **`-libjars`** parameter.

Special characters support in Hive

NPS supports UTF-8 special characters for the table names, while Hive only allows alphanumeric characters and underscores. If this leads to problems during data movement, change the names of the NPS tables to be compatible with Hive.

Using JDBC to connect to the Hive server

By default, IBM Fluid Query connects to Hive directly in order to create table metadata. If for some reason you are unable to establish a direct connection, you can configure IBM Fluid Query to use JDBC for connections to Hive.

This remote import option is an alternative to the standard import. The procedure for running remote import is very similar to running standard import and export:

1. First, you must configure an XML configuration file, in this case `fq-import-remote-conf`. The file can be found in the `fluid-query-import-export-<version>.tar` package that you extracted as part of the installation process. It contains additional properties that are used for connecting to Hive using JDBC.
2. To perform remote import, run the following command:

```
hadoop jar /fluidqueryLocal/nzcodec.jar -conf fq-import-remote-conf -libjars /fluidqueryLocal/nzcodec.jar,/fluidqueryLocal/nzjdbc3.jar
```

Note: In remote import, you do not need to export the classpath for the Hadoop job which is a prerequisite in the standard import procedure.

Make sure you provide all the required properties in the configuration file. The default values are already set:

Hive JDBC connection properties

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

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

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

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

-
  ```
  <property>
   <name>nz.fq.sql.type</name>
   <value>hive1</value>
  </property>
  ```

  The **nz.fq.sql.type** property sets the server type. Supported types are **hive1** or **hive2**. The recommended one is **hive2**.

Hive JDBC authentication properties

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

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

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.password</name>
   <value>passw0rd</value>
  </property>
  ```

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

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

  The **nz.fq.sql.spn** property sets the Kerberos service principal name. It is required if you want to use Kerberos authentication. Sample value: **hive/horton-master.ibm.com@XXXXXX.IBM.COM**

Hive JDBC SSL properties

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

  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.sslTrustStore</name>
   <value>$HIVE_HOME/src/data/files/cacerts_test.jks</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.sslTrustStorePassword</name>
   <value>passw0rd</value>
  </property>
  ```

  The **nz.fq.sql.sslTrustStorePassword** property sets the password to the specified SSL trustStore.

General configuration

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

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

-
  ```
  <property>
   <name>nz.fq.clean.before.import</name>
   <value>true</value>
  </property>
  ```

  The **nz.fq.clean.before.import** property defines whether all files that start with `part-0000` are to be removed from the target folder that is specified in property **nz.fq.output.path** before running the import. It prevents the conflict that occurs when performing multiple imports to the same directory. It does not affect directories, only files.

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

  The **nz.fq.sql.metadata** property defines whether IBM Fluid Query will create the table in Hive. By default, it is set to **true**. If you set it to **false**, the table will not be created in Hive and IBM Fluid Query will only import data files and put them in HDFS.

HDFS properties

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

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

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

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

-

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

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

Compression properties

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

  The **nz.fq.compress** property defines whether to transfer NPS data in compressed internal format.

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

  The **nz.fq.output.compressed** property defines whether the transferred data is stored in Hadoop in compressed internal format. Depends on the **nz.fq.compress** setting.

NPS properties

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

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

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

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

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

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

-
  ```
  <property>
   <name>nz.fq.nps.port</name>
   <value>5480</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.user</name>
   <value>admin</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.password</name>
   <value>password</value>
  </property>
  ```

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

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

The **nz.fq.nps.ssl.cacertificate** property sets the CA Certificate file used to authenticate connections. Used only when the SSL flag is true. If not provided, then all connections will be accepted.

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

The **nz.fq.nps.where** property specifies the SQL WHERE clause that is used for selecting the data to transfer.

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

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

Configuring connection to NPS with Kerberos authentication

If you want to perform import and export operations using Kerberos authentication, you must first perform some manual configuration steps.

1. Create the `login.config` file on all Hadoop nodes 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
        debug = true;
     };
     ```

   - For BigInsights (IBM Java):

     ```
     EntryModuleName{
        com.ibm.security.auth.module.Krb5LoginModule required
        debug = true;
     };
     ```

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.
4. Edit the `/etc/krb5.conf` file so that it reflects the actual Kerberos setup. You can copy this file from NPS.

Run these steps to check whether the Kerberos authentication is configured correctly:

1. Copy the `nzjdbc3.jar` file to `/fluidqueryLocal` on the Hadoop node.
2. Run the following command on Hadoop:

```
/usr/bin/java -Djava.security.auth.login.config=login.conf -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.

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® 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 (® or ™), 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.

Index

Part Number: 29000 Rev. 1

Printed in USA