pitney bowes

# Spectrum™ Technology Platform

Version 12.0 SP2

## Data & Address Quality for Big Data SDK Guide

# Table of Contents

# 1 - Getting Started

## In this section

# Introduction

The Spectrum™ Data & Address Quality for Big Data SDK helps you create, configure and run MapReduce jobs, Spark jobs, and Hive User-Defined Functions for Data Quality operations on a Hadoop platform.

Using the SDK, you can create and run the jobs directly on a Hadoop platform, thus eliminating network delays and running distributed Data Quality processes in cluster, resulting in a remarkable improvement in the performance.

> **Note:** You can also use Amazon S3 Native FileSystem (s3n) as input and output for Hadoop MapReduce and Spark jobs.

### *SDK Usage*

This SDK can currently be used through Java APIs and Hive User-Defined Functions (UDFs).

- Java APIs
  - MapReduce API
  - Spark API
- Hive User-Defined Functions

# Who should use the SDK?

The Spectrum™ Data & Address Quality for Big Data SDK is intended for:

1. Customers who want to check the data quality of the data residing on hadoop.
2. Hadoop developers familiar with MapReduce or Spark programming who wish to create a solution around a certain use case.
3. Hadoop developers who want to perform *data cleansing*, *data enriching*, *data deduplication*, and *data consolidation* operations over existing data.
4. Hive users who are not familiar with the complexities of MapReduce or Spark but are comfortable with Hive Query Language (HQL), which is syntactically similar to SQL.

# Workflow

To use the SDK, these components are required:

**Spectrum™ Data & Address Quality for Big Data SDK Installation**
The Spectrum™ Data & Address Quality for Big Data SDK JAR file must be installed on your system and available for use by your application.

**Client Application**
The Java application you must create to invoke and run the required Data Quality operations using the SDK. The Spectrum™ Data & Address Quality for Big Data SDK JAR file must be imported into your Java application.

**Hadoop Platform**
On running a job using the Spectrum™ Data & Address Quality for Big Data SDK, data is first read from the configured Hadoop platform, and after relevant processing, the output data is written to the Hadoop platform. For this, the access details of the Hadoop platform must be configured correctly in your machine. For more information, see **Installing SDK on Windows** on page 11 and **Installing SDK on Linux** on page 12.

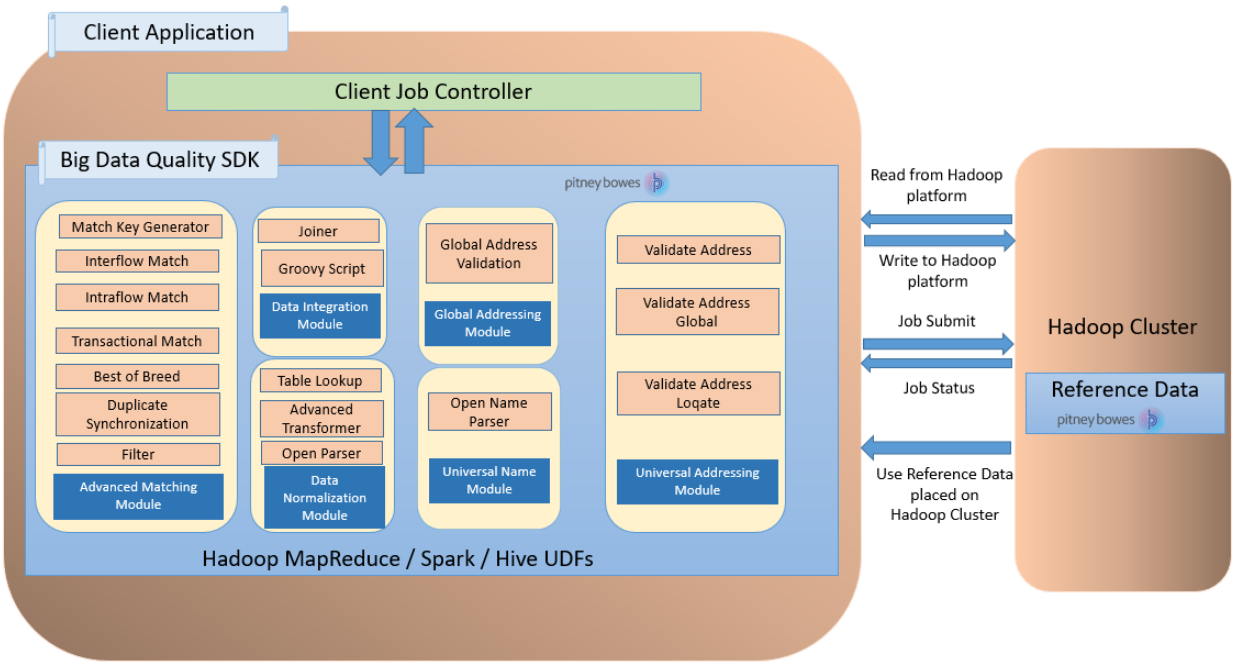
> **Note:** You can also use Amazon S3 Native FileSystem (s3n) as input and output for Hadoop MapReduce and Spark jobs.

**Reference Data**
The Reference Data, required by the Spectrum™ Data & Address Quality for Big Data SDK, is placed on the Hadoop cluster.

**Java API and Hive UDF, UDAF**
To use the Java API, Hive UDF or Hive UDAF, you can opt to place the reference data on Local Data Nodes or Hadoop Distributed File System (HDFS) :

- **Local Data Nodes**: The Reference Data is placed on all available data nodes in the cluster.
- **Hadoop Distributed File System (HDFS)** The Reference Data is placed on an HDFS directory and while running the jobs you can specify if data is to be downloaded as distributed cache or to a local directory. For more information, see **Placement and Usage of Reference Data** on page 16.

**Note:** The SDK also enables *Distributed Caching* for enhanced performance.




Big Data Quality SDK- Workflow

# Modules and Jobs

The modules supported in the Spectrum™ Data & Address Quality for Big Data SDK and the jobs you can run in each of these modules are:

| Modules | Jobs supported through Java APIs and Hive UDFs |
|---|---|
| Advanced Matching Module | **Match Key Generator job** on page 73 |
| | **Interflow Job** on page 55 |
| | **Intraflow Job** on page 65 |
| | **Transactional Match Job** on page 79 |
| | **Best of Breed Job** on page 36 |
| | **Duplicate Synchronization Job** on page 42 |
| | **Filter Job** on page 48 |
| Data Integration Module | **Joiner Job** on page 92<br>**Note:** This job can be performed only through Java APIs. |

| Modules | Jobs supported through Java APIs and Hive UDFs |
|---|---|
| | **Custom Groovy Script Job** on page 87 |
| Data Normalization Module | **Table Lookup Job** on page 108 |
| | **Advanced Transformer Job** on page 97 |
| | **Open Parser Job** on page 104 |
| Global Addressing Module | **Global Address Validation** on page 114 |
| Universal Addressing Module | **Validate Address** on page 124 |
| | **Validate Address Global** on page 138 |
| | **Validate Address Loqate** on page 149 |
| Universal Name Module | **Open Name Parser** on page 159 |

# Reports

You can generate reports using the Spectrum™ Data & Address Quality for Big Data SDK for these jobs:

- Interflow Match
- Intraflow Match
- Transactional Match
- Global Address Validation
- Open Name Parser
- Validate Address
- Validate Address Global
- Validate Address Loqate

The report is generated in the form of counters which tracks the number of duplicate records, the number of unique records, and other useful parameters for an executed job. You can choose to view the report directly on the console or save it as a file locally.

# 2 - Installing the SDK

## In this section

# System Requirements

*For Hadoop Distributed File System (HDFS):*

1. Java JDK version 1.7 and above.
2. Hadoop version 2.6 and above
3. Spark 2.0.1 and above.

*For Hive:*

1. Hive version 1.2.
2. A Hive client of your choice. For example, Beeline.

**Note:** Spectrum™ Data and Address Quality for Big Data SDK can be run only with Hadoop clusters.

# Operating System Updates

Before installing the Spectrum™ Data & Address Quality for Big Data SDK, be sure to apply all the latest product updates available for your operating system, especially those that resolve issues with Java.

# Installer Inclusions

The SDK installation ZIP file contains these components:

1. `Readme.txt`
2. `sdkinst.bin`: Installer for LINUX machines.
3. `sdkinst.exe`: Installer for WINDOWS machine.

# Installing SDK on Windows

To install the Spectrum™ Data & Address Quality for Big Data SDK on a Windows machine, follow these steps:

1. Download the Spectrum™ Data & Address Quality for Big Data SDK ZIP installer file using the download instructions in your welcome email or the release announcement email.

   **Note:** The name of a typical installer ZIP file is: *BigDataSDK120F0101.zip*.

2. Extract the contents of the downloaded zip file to a location where you want to install Spectrum™ Data & Address Quality for Big Data SDK.
3. Go to the installation directory and locate the installer named *sdkinst.exe*.
4. Double-click `sdkinst.exe`. The installation wizard appears.
5. Click **Next**. The **Choose Install Folder** window appears.

   Here, you can specify the folder where you want to install Spectrum™ Data & Address Quality for Big Data SDK. For example, `C:\Program Files\Pitney Bowes\Spectrum BigDataSDK\SDK`.

   a) Click the **Choose** button to select the required folder.
   b) Click the **Restore Default Folder** button to select the default folder.
6. Click **Next**.

   In the **Pre-Installation Summary** screen, review the installation information.
7. Click **Install**.
   The Spectrum™ Data & Address Quality for Big Data SDK is installed on your computer
8. Click **Done**.
9. Verify that you have set up the SDK correctly. Go to the location where you have installed the SDK, for example `C:\Program Files\Pitney Bowes\Spectrum BigDataSDK\SDK`.

Once you have successfully installed the SDK on your machine, these folders are added in the install directory:

- `API`
- `Documentation`
- `modules`
- `samples`
- `utilities`

**Note:** To use the jobs of Data Normalization Module, Global Addressing Module, Universal Name Module or Universal Addressing Module, you must install the respective Reference Data for each module.

# Installing SDK on Linux

To install the Spectrum™ Data & Address Quality for Big Data SDK using command line on a Linux machine, follow these steps:

1. Download the Spectrum™ Data & Address Quality for Big Data SDK using the download instructions in your welcome email or the release announcement email.
2. Extract the contents of the downloaded file to a location where you want to install Spectrum™ Data & Address Quality for Big Data SDK.
3. Change the directory to the location.
4. Ensure you have `execute` permission on the files by typing this command:

   ```
   chmod a+x sdkinst.bin
   ```

5. Run this command:

   ```
   ./sdkinst.bin
   ```

   Follow the command prompt instructions.

6. When prompted, provide the directory where you want to install the SDK.
   For example, `/home/hadoop/BDQ_InstallPath`.

   **Note:** If you select a non-default folder as the installation directory, ensure that the length of the absolute installation path does not exceed 34 characters.

   The default installation path with 27 characters is admissible:

   ```
   /root/PBSpectrum_BigDataSDK
   ```

   A pre-installation summary is displayed.

7. Review the summary and press `ENTER` to continue with the installation.
8. See the installation log file to verify that the Spectrum™ Data & Address Quality for Big Data SDK has been installed correctly.
9. When you are done, press `ENTER` to finish and exit the installer.

Once you have successfully installed the SDK on your machine, these folders are added in the install directory:

- `API`
- `Documentation`
- `modules`
- `samples`
- `utilities`

**Note:** To use the jobs of Data Normalization Module, Global Addressing Module, Universal Name Module or Universal Addressing Module, you must install the respective Reference Data for each module.

## Running Acushare Service

Before creating and running the first *Validate Address* job, you must run the Acushare service on each node of the Hadoop or Spark cluster.

**Note:** This is a one-time mandatory activity to be performed before running the first *Validate Address* job.

On each node of the cluster:

1. Copy the Acushare setup script `sdkrts.bin` from the Spectrum™ Data & Address Quality for Big Data SDK installation path to any location on the node.

   **Note:** On the SDK server, the Acushare setup script `sdkrts.bin` is in `<BDQ SDK_InstallPath>/SDK/utilities/dbloader/aq/runtime/bin`.

2. In the `installer.properties` file (located in the same directory as the Acushare setup script), specify the `USER_INSTALL_DIR` details. This is the path where you want to install the Acushare service.

   **Important:** The path should not be more than 60-characters-long.

   The Acushare service starts automatically once the installation completes successfully.
3. Alternatively, to start the Acushare service manually on a node, go to `<Acushare runtime path>/runtime` and run the script file `startrts.sh` with the argument `<Acushare runtime path>/runtime`.

| | |
|---|---|
| **Stopping Acushare service** | To stop the Acushare service on any node, go to `<Acushare runtime path>/runtime` and run the script file `stoprts.sh` with the argument `<Acushare runtime path>/runtime`. |
| **Uninstalling Acushare service** | To uninstall the Acushare service from any node, run the script file `Uninstall_SDKRTS.sh` placed at `<Acushare runtime path>/Uninstall`. |

# 3 - Using Reference Data

## In this section

# Reference Data Overview

The Pitney Bowes Reference Data defines a set of permissible values to be used by other data fields in your system to ensure data quality. It enhances data validity, accuracy and consistency. It enables you to extract more value from your data and obtain trusted data from Big Data system.

For example, if you use the Reference Data with Data Normalization Module, you can establish a single customer identity across the enterprise. A well-defined customer information is the first step towards improving operational efficiency.

The reference data is updated periodically on the e-Store, from where you need to fetch it and install it at one of these locations:

- All the data nodes of Hadoop cluster
- Hadoop Distributed File System (HDFS)

## Scripts to install reference data

The scripts to install reference data resides in the `Utilities/dbloader` folder of the SDK installation directory. The child folders at this location are:

**dataquality** JAR and scripts to install the Reference Data for:

- Data Normalization Module
- Universal Name Module

> **Note:** For more information, see **Extraction through interactive utility** on page 21 and **Extraction using silent script** on page 22.

**aq**

- The `scripts/server/installdb_unc.sh` and `scripts/server/silentInstalldb_unc.sh` scripts to install the Reference Data. You must run this script to install or extract the data.
- `runtime` folder containing Acushare service set-up information for Universal Addressing Module's *Validate Address* job.

> **Note:** For more information, see **Extraction through interactive utility** on page 21 and **Extraction using silent script** on page 22.

# Placement and Usage of Reference Data

You can place the reference data at one of these locations:

- All the data nodes of Hadoop cluster
- Hadoop Distributed File System (HDFS): If your reference data is on HDFS, you have these two options for managing it when you run the jobs:
  - **Download it to the current working directory**

    The reference data gets downloaded to your working directory as temporary files. Every time your job is completed, these files are deleted from the working directory, making fresh download of reference data mandatory for each job.
  - **Download data to a local path**

    The reference data is downloaded to a local data path you specify, and it remains available for all the jobs till the data gets refreshed on HDFS.

### *Managing reference data on HDFS*

For successfully downloading reference data from HDFS to a specified local path and to be able to run jobs using that reference data, you need to ensure:

- The user executing the jobs has write access to the local drive on each data node.
- There is sufficient disk space on each of the data nodes to download data from HDFS.

### *Advantages of downloading data to a local path*

- You do not need to place data on each of the nodes as you do in case reference data is copied to or placed on all the nodes (Localtodatanodes option).
- On any given data node, same version of data is downloaded only once.
- There is no limit to reference data download.

### *Properties to be specified in jobs*

These are the properties you need to specify in the job configuration files to indicate the chosen reference data strategy and path.

- **Reference data is placed on all the data nodes of Hadoop cluster**:

  In the Json string, specify these details:

  - *referenceDataPathLocation*: `LocaltoDataNodes`
  - *dataDir*: Path where the reference data is located.

```
<property>
        <name>pb.bdq.reference.data</name>
```

```
        <value>{"referenceDataPathLocation":"LocaltoDataNodes",
               "dataDir":"/home/data/referenceData"}</value>
        <description>Pass reference data details as JSON
format.</description>
</property>
```

- **Reference data is placed on Hadoop Distributed File System (HDFS) and you want to use the distributed cache mode:** In the Json string, specify these details:
  - *referenceDataPathLocation*: `HDFS`
  - *dataDir*: Path of the reference data on HDFS
  - *dataDownloader*: `DC`

```
 <property>
        <name>pb.bdq.reference.data</name>
        <value>{"referenceDataPathLocation":"HDFS",
        "dataDir":"./referenceData",
        "dataDownloader":{"dataDownloader":"DC"}}</value>
        <description>Pass reference data details as JSON format.
        Pass above format for DATA DOWNLOADER when data is in
HDFS</description>
 </property>
```

- Reference data is placed on Hadoop Distributed File System (HDFS) and you want to download it to a local path:

  In the Json string, specify these details:

  - *referenceDataPathLocation*: `HDFS`
  - *dataDir*: Path of the reference data on HDFS
  - *dataDownloader*: `HDFS`
  - *localFSRepository*: Path where the reference data needs to be downloaded locally.

```
<property>
        <name>pb.bdq.reference.data</name>
        <value>{"referenceDataPathLocation":"HDFS",
         "dataDir":"/home/data/dm/referenceData",
         "dataDownloader":{"dataDownloader":"HDFS",
         "localFSRepository":"/local/download"}}</value>
        <description>Pass reference data details as JSON
format.</description>
</property>
```

# Extracting Reference Data for DNM, UNM Jobs

To extract the Reference Data for **Data Normalization Module** and **Universal Name Module** jobs you need to run the data loader script, for example `installdb_dnm`.

Ensure the script file (for example `installerdb_dnm`), and the JAR file reside in the same folder.

1. Log in to your machine.
2. Change the directory to the location where you have installed the SDK.

   After you have successfully installed the Spectrum™ Data & Address Quality for Big Data SDK on your machine, you should have the Reference Data loader in the directory `BDQ_InstallPath/SDK/utilities/dbloader/unix/bin`.

3. Run the reference data loader script. (For example, `installdb_dnm`).
   A numbered list of stages is displayed and you are prompted to select the stage.
4. Type the number corresponding to the stage for which you want to load the data.
5. Specify the path where you want the reference data sets to be extracted and placed after download.

   The reference data input are the base tables of Data Normalization Module, core name data bases required to perform the Data Normalization and Universal Name Module jobs.

6. Specify the path for the output directory.
7. The system prompts whether you want to view the log file. Select as desired.
8. The system starts loading the data. The data is extracted in the specified output directory.
9. Repeat the steps for each stage.

Reference data is now available to be used.

**Note:** You must have `write` permission to the *reference data path* to run the jobs.

## User Defined Reference Data

You can use your input data files as reference data along with the preshipped Pitney Bowes reference data. Your input files must have rows with **semicolon** separated keys and values. To convert your input files into the Pitney Bowes reference data format, execute this command on the console:

```
java -cp [dataquality-12.2-jar-with-dependencies.jar] UserLibraryCreater
[-input <arg>] [-output <arg>]
```

Here, `-input <arg>` is the path where you have placed your input files and `-output <arg>` is the path where converted input files will be placed.

**Note:** You have to place your converted input files suffixed with *-user* for example, *cdq-AdvTransformer-user.db* and *cdq-AdvTransformer-user.lg* to the path where DNM,UNM reference data was previously extracted.

# Extracting Reference Data for GAM Jobs

This section describes the process for fetching and extracting the reference data sources for **Global Addressing Module** job. The reference data files are placed at e-Store in SPD format.

1. Fetch SPD files from the e-Store. Each SPD file has reference data for a specific country or region. A sample set of GAV reference data is given below.

| SPD File Name | SPD Description | |
|---|---|---|
| GAC062017.spd | China GAV data | GAV_CHINA_JUN2017 |
| GAA062017.spd | Americas GAV data | GAV_AMERICAS_JUN2017 |
| GAE062017.spd | EMEA GAV data | GAV_EMEA_JUN2017 |
| GEP062017.spd | EMEA Premium GAV data | GAV_EMEA2_JUN2017 |
| GAP062017.spd | APAC GAV data | GAV_APAC_JUN2017 |
| GAW062017.spd | World GAV data | GAV_WORLD_JUN2017 |
| GGR062017.spd | Germany AddressPoint GAV data | GAV_GERMANY_ADDRESSPOINT_JUN2017 |
| GES062017.spd | Spanish TT GAV data | GAV_SPANISH_JUN2017 |
| GBR062017.spd | Great Britain GAV data | GAV_GREATBRITAIN_JUN2017 |
| GGB062017.spd | GBR AB Subscription GAV data | GAV_GBR_AB_JUN2017 |
| GIN062017.spd | India (Domestic) GAV data | GAV_INDIA_JUN2017 |
| GMX062017.spd | Mexico NAVTEQ GAV data | GAV_MEXICO_NAVTEQ_JUN2017 |
| GSE062017.spd | Sweden GAV data | GAV_SWEDEN_JUN2017 |
| GAU062017.spd | Australia GNAF GAV data | GAV_AUSTRALIA_GNAF_JUN2017 |

**Note:** The numbers in the SPD file name denote month and year of the reference data. For example in `GAC062017`, `06` denotes the sixth month and `2017` is the year.

2. To use reference data on HDFS, place the SPD files on HDFS and use the path while running the job.
3. To use reference data on local data nodes of Hadoop clustser, extract the SPD files on a local directory using the command:

```
unzip <spd file name> -d <directory to extract>
```

**For example,**

`unzip GAC062017 -d /home/hadoop/hduser/GAM_Feb_2018_DB/databases`

where, `GAC062017` is the SPD file name and `/home/hadoop/hduser/GAM_Feb_2018_DB/databases` is the directory where you want the reference data to be extracted.

# Extracting Reference Data for UAM Jobs

This section describes the process for fetching and extracting various reference data sources for validate address jobs. The reference data is fetched from the e-Store.

> **Note:** For the Validate Address and Validate Address Global jobs, the Reference data must be placed on all the data nodes of Hadoop cluster or Hadoop Distributed File System (HDFS). For the Validate Address Loqate job, it must be placed at one node and that further needs to be mounted to all other data nodes.

### *Validate Address Loqate*

1. Fetch reference data from the e-Store.
2. Extract the contents of the *ZIP* file.
3. Place the extracted files on one node and mount it further to all the other data nodes.

The files are now ready to be used in different map reduce and spark jobs and user defined functions.

### *Validate Address Global - Address Doctor*

1. Fetch the reference data from the e-Store. In case of *Validate Address Global*, the reference data is available in these six data bundles:
   - UAM - Enhanced International - Americas - Bundle Data
   - UAM - Enhanced International - Americas - Bundle Data 2
   - UAM - Enhanced International - EMEA - Bundle Data
   - UAM - Enhanced International - EMEA - Bundle Data 2
   - UAM - Enhanced International - APAC - Bundle Data
   - UAM - Enhanced Int - US Cert Subscription
2. Download the *ZIP* files and place those on HDFS. Do not extract the *ZIP* files for placing on HDFS.

   > **Note:** To place the reference data on local nodes, extract the zip files and place it on all the data nodes. The path needs to be same for all the data nodes.

The files are now ready to be used in different map reduce and spark jobs and user defined functions.

### *Validate Address - C1P*

1. Fetch these reference data bundles from the e-Store.
   - US_SUB
   - DPV
   - EWS

- LACS
- SUITE

2. Extract the data through
   - An interactive utility using the script `installdb_unc.sh` (See **Extraction through interactive utility** on page 21)
   - A silent script `silentInstalldb_unc.sh`. (See **Extraction using silent script** on page 22)

The data gets extracted to the local or edge node, from where, it can be pushed to HDFS to be used in map reduce and spark jobs and Hive user defined functions.

## Extraction through interactive utility

**Note:** Ensure that `execute` permission is granted to the `aq` folder.

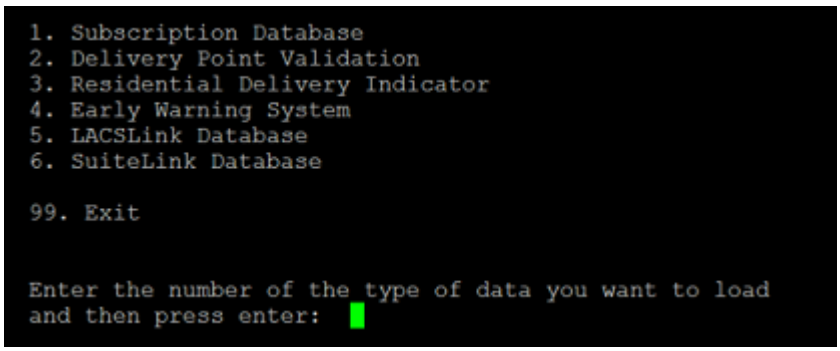
1. Log in with admin rights or as a root user.
2. Change the directory to the location `<BDQ_Installation>/SDK/utilities/dbloader/aq/scripts/server`.
3. Run the script `installdb_unc` using the command:

   ```
   sh installdb_unc.sh <BDQ_Installation/SDK> <Acushare runtime path>
   ```

   This command also verifies whether the Acushare service is running and starts the service if it is not already running. It also displays these options:

   - **US Subscription**: Press 1 to list the available types of data loading.
   - **Exit**: Press 99 to exit.

4. Enter the specific number for the type of data you want to load.




5. Fetch reference data from the e-Store, unzip it, and place it in a folder. Path up to this folder is your input path. For example, `/home/hadoop/hduser/UAM_DEC_2017_DB/databases`.
6. Specify the input path.
   The utility displays a default output path.

7. Enter `c` to continue, `m` to modify the default path or `q` to quit.

```
The Residential Delivery Indicator load environment is currently set to:

Residential Delivery Indicator input file location:
Residential Delivery Indicator output file location: /root/SDK/utilities/dbloader/addressquality/

Enter c to (c)ontinue
   or m to (m)odify
   or q to (q)uit

   ===>
```

   The input data is extracted at your designated output file location.
8. The system prompts to verify whether or not your new RDI file location is correct. Enter `y` or `n`.

```
Please enter full path where you would like to install
the RDI file ==> /root/out

The new RDI file location will be:  /root/out
Is this correct?

Enter (y)es to continue.
      (n)o to try again.

   ===> y

The RDI file output location /root/out does not exist.
Do you want to create it now?

Enter (y)es to create the new RDI file area.
      (n)o to exit.

   ===>
```

The system starts loading the data. The data is extracted in the specified output directory.

**Note:** Repeat these steps for the type of data that you want to load.

## Extraction using silent script

Use `silentInstalldb_unc.sh` script for extracting the reference data without going through an interactive process. The script accepts arguments once and extracts databases silently on your machine.

**Note:** Ensure that `execute` permission is granted to the `aq` folder.

1. Log in with rights or as a root user.
2. Change the directory to the location to:`<BDQ_Installation>/SDK/utilities/dbloader/aq/scripts/server`.
3. Run the script `silentInstalldb_unc.sh` using the command:

```
./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path>
```

This command also verifies whether the Acushare service is running. If not, then this command starts the service.

4. This table describes the command.

| Required | Argument | Description |
|---|---|---|
| Yes | `-i` *-input* | Path of the input data to be extracted. This should be the path till the folder in which you have the input data. For example, `/home/hadoop/hduser/UAM_DEC_2017_DB/databases` |
| Yes | `-o` *-output* | Path of the output location for extracted data |
| Yes | `-d` *-database* | Database type to be extracted. The values are:<br>• US_SUB<br>• RDI<br>• EWS<br>• LACS<br>• DPV<br>• SUITE |
| Yes | `-a` *-acushare* | Path of the installed acushare service |
| No | `-z` *-zip* | To create output files in the `tar.gz` format.<br>Pass the value `Y` to create a compressed `tar.gz` output. Default is `N`. |
| No | *-override* | To override the output location. Pass the value `N` if you do not want to override the output location. Default is `Y`. |
| No | *-optionaldb* | To load optional EOT file while extraction. Pass the value `N` if you do not want to load an optional EOT file. Default is `Y`.<br>**Note:** Valid only for the US_SUB database. |
| No | *-ewsfile* | Specify the name of the EWS file. Default is `OUT`<br>**Note:** Valid only for the EWS database. |

These are specific commands to extract specific database types:

- US_SUB: `./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path> -override N -optionaldb N -zip Y`
- DPV: `./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path> -override N -zip Y`
- EWS: `./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path> -override N -ewsfile <name of file> -zip Y`
- LACS: `./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path> -override N -zip Y`
- SUITE: `./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path> -override N -zip Y`

> **Note:** Run the help command `./silentInstalldb_unc.sh -help` to view the details of mandatory and optional parameters.

# 4 - The Java API

## In this section

# About Java

A Java *class* is a blueprint or prototype that defines the variables and methods common to all instances of a certain type. It defines the implementation of a particular kind of instance.

A Java *object* is an instance of a Java class. It is a real time instance of Java classes, created using the Java Virtual Machine. An instance of a class, handled using a variable, encapsulates the real time information of the class.

*Methods* of a class define the various functions a class or its object must perform. Methods are similar to the functions or procedures in procedural languages such as C.

*Parameters* are used to pass the information an object requires to perform a certain task.

Java software objects interact and communicate with each other using *messages*.

For more information about Java technology, see **www.oracle.com/java**.

# Components of the SDK Java API

The key components to use a Spectrum™ Data & Address Quality for Big Data SDK job using the Java API are:

**JAR Files**

1. Hadoop JAR files.
2. The JAR files of the module to which the desired Spectrum™ Data & Address Quality for Big Data SDK job belongs, as indicated in the table:

| Module | Job | JAR File |
|---|---|---|
| Advanced Matching Module | All AMM jobs | *amm.core-12.2.jar* |
| Data Integration Module | All DIM jobs | *dim.core-12.2.jar* |
| Data Normalization Module | All DNM jobs | *dnm.core-12.2.jar* |
| Global Addressing Module | Global Address Validation | *gam-global addressvalidation.core-12.2.jar* |
| Universal Addressing Module | Validate Address | *uam-universaladdress.core-12.2.jar* |

| Module | Job | JAR File |
|---|---|---|
| Universal Addressing Module | Validate Address Global | *uam-global.core-12.2.jar* |
| Universal Addressing Module | Validate Address Loqate | *uam-loqate.core-12.2.jar* |
| Universal Name Module | All UNM jobs | *unm.core-12.2.jar* |

**Configuration Files** Files in XML format containing all parameters and values required to run a job, including match rules, input file details, output file details, and MapReduce or Spark configuration details.

Sample configuration XML files are placed at the location `<Big Data Quality bundle>\samples\configuration`.

**Client Java Application** Java application to use the API to create and run the required Spectrum™ Data & Address Quality for Big Data SDK job provided by its Java API.

**Hadoop Platform** The created job accesses the configured Hadoop platform to access input data and dump the output data in a file.

# Using the Software Development Kit

The SDK can be used to run Spectrum™ Data & Address Quality for Big Data SDK jobs using any one of these two approaches:

1. On a console, directly run the module-specific JAR files and pass the various XML-format configuration properties files as arguments to the commands.

   For MapReduce jobs run the `hadoop` command, while for Spark jobs run the `submit-spark` command.

   For the steps, see **Using Configuration Property Files** on page 170.

2. Create your own Java client project by importing the relevant Spectrum™ Data & Address Quality for Big Data SDK module JAR file, specify all required job configurations for your desired job within your client project and run it.

   For the steps, see **Creating a Java Application** on page 28.

## Creating a Java Application

Ensure the Spectrum™ Data & Address Quality for Big Data SDK is installed on your machine.

To use the SDK:

1. Create a Java project to use the SDK as required using one of these methods:
   a) Create a specific Java project to run the required Data Quality operation.

      Using this method, you'll need to create separate Java projects for each Data Quality job you wish to run.

   b) Create a common Java project to run any of the desired Data Quality operations using the corresponding runtime arguments.

      Using this method, you'll need to create just one Java project which accepts runtime arguments corresponding to the desired Data Quality operation.
2. Import the Spectrum™ Data & Address Quality for Big Data SDK module-specific JAR file into your project to use the SDK. For a list of the module-specific JAR files, see **Components of the SDK Java API** on page 26.
3. Import the required Hadoop JAR files into your project.
4. Create your application to run the desired Data Quality jobs, with appropriate configurations.
5. Build your project, using any build tool like Maven or Ant.
   A JAR file of your project is created as a result.

   For example, `MatchKeyGeneratorClient-with-dependencies.jar` is created.
6. Place your project's JAR file on the Hadoop platform.
7. On the Hadoop platform, in a command prompt, change the directory to the path where you have placed your JAR file.
8. Run the JAR of your project using the command:

```
hadoop jar <name of the JAR of your client project> <fully qualified
 name of the main class>
```

   For example:

```
hadoop jar MatchKeyGeneratorClient-with-dependencies.jar
com.company.bdq.amm.mr.MatchKeyGeneratorJob
```

The desired job is created and executed on the Hadoop platform.

Your Java application accesses the input data from the path specified on the Hadoop platform, and creates and runs the job on the Hadoop platform. The output of the job is dumped into a file at the specified output path on the Hadoop platform.

# Common API Entities

## ConjoinedRule

*Purpose*
A type of consolidation rule, which is used when multiple rules are to be joined using `AND` and `OR` operators. A conjoined rule can include simple rules as its components. See **SimpleRule** on page 32.

This class allows defining rules for the **Advanced Matching Module** and the **Data Normalization Module** jobs.

## ConsolidationCondition

*Purpose*
To specify the consolidation rules and the corresponding action for the **Advanced Matching Module** and the **Data Normalization Module** jobs.

### ConsolidationRule

*Purpose*
To specify the consolidation rule based on which it must be determined whether action is required on a record or not.

This class allows defining consolidation rules for the **Advanced Matching Module** and the **Data Normalization Module** jobs.

### ConsolidationAction

*Purpose*
To specify the field which must be copied to other records in a group for a particular consolidation condition.

This class allows defining consolidation actions for the **Advanced Matching Module** and the **Data Normalization Module** jobs.

# FilePath

*Purpose*

To specify the details of an input and output text file to run a job.

**Table 1: The parameters and values of FilePath sub-class**

| Parameters | Values | Data type |
|---|---|---|
| path | path of the input and output text file.<br>In case your files are placed on Amazon S3 Native FileSystem (s3n), you need to enter the path in this format:<br>`s3n://AWS Access key ID: AWS Secret access key@<Bucket name followed by input file path>`<br>`s3n:// AWS Access key ID: AWS Secret access key@<Bucket name followed by output file path>` | Path |
| recordSeparator | Record separator | String |
| fieldSeperator | Field separator | String |
| fileHeader | Header fields | String |

# JobConfig<T extends ProcessType>

*Purpose*

An interface to specify Hadoop configurations for a job.

## MRJobConfig

*Purpose*

To specify Hadoop configurations for any MapReduce job.

### SparkJobConfig

*Purpose*
To specify Hadoop configurations for any Spark job.

## JobDetail<T extends ProcessType>

*Purpose*
Stores the basic information needed for creation of a job.

## JobFactory

*Purpose*
The base interface to specify to create job instances and specify the details of the jobs to be created.

## JobPath

*Purpose*
The parent class to specify the details of input source and output destination for a job.

## OrcFilePath

*Purpose*

To specify the input or output paths of ORC format files to run a job.

The `path` parameter takes input and output file path as values.

## ParquetFilePath

*Purpose*

To specify the input or output paths of Parquet format files to run a job.

The `path` parameter takes input and output file path as values.

## ProcessType

*Purpose*
The parent markup interface for all supported process types, like MapReduce and Spark.

### MRProcessType

*Purpose*
To specify the MapReduce process type for jobs.

### SparkProcessType

*Purpose*
To specify the Spark process type for jobs.

## ReferenceDataPath

*Purpose*
To specify the path of the Reference Data for a job.

## ReportManager

*Purpose*
An interface for retrieving the reporting statistics of a job.

## SimpleRule

*Purpose*
A type of consolidation rule. A simple rule can be used alone and as a component of a conjoined rule. See **ConjoinedRule** on page 29.

## Exceptions

### JobException

*Purpose*
Handles job-specific exceptions, displaying appropriate messages.

# Advanced Matching Module Jobs

## Common Module API

### AdvanceMatchDetail<T extends ProcessType>

*Purpose*
To specify the details of an **Advanced Matching Module** job.

### AdvanceMatchFactory

*Purpose*
A singleton factory class to create instances of **Advanced Matching Module** jobs.

### GroupbyOption<T extends ProcessType>

*Purpose*
To specify the column on which grouping is to be performed for an Advanced Matching job.

### *GroupbyMROption*

*Purpose*
To specify the column on which grouping is to be performed for an Advanced Matching MapReduce job.

### *GroupbySparkOption*

*Purpose*
To specify the column on which grouping is to be performed for an Advanced Matching Spark job.

### MatchKeySettings

*Purpose*
Maintains a `List` of match keys for a Match Key Generator job.

### MatchRule

*Purpose*
Allows creation of matching rules for Advanced Matching jobs.

This is done by defining a hierarchy of parent and child nodes. Each node maps to one of the input fields to be matched.

#### *ChildMatchRule*

*Purpose*
To specify a child node of a match rule, which maps to a field and certain algorithms and other properties.

#### *ParentMatchRule*

*Purpose*
To specify a parent node of a match rule, which is a logical grouping of other parent nodes and child nodes.

## Special Scenarios

*Records with Blank Group-By Column*
All records with a blank group-by value are marked as malformed records, and dumped in separate files in the output HDFS folder.

These malformed files are named as:

| | |
|---|---|
| **Malformed Records in Candidate Files** | Candidate file records with a blank group-by column are discarded as malformed records and inserted into files with the file naming convention `malformedRecordsCandidate-m-<5 digit numeral>`.<br><br>For example, `malformedRecordsCandidate-m-00000`, `malformedRecordsCandidate-m-00001`.<br><br>This applies to Interflow Match jobs. |

| | |
|---|---|
| **Malformed Records in Suspect Files** | Suspect file records with a blank group-by column are discarded as malformed records and inserted into files with the file naming convention `malformedRecordsSuspect-m-<5 digit numeral>`.<br><br>For example, `malformedRecordsSuspect-m-00000`, `malformedRecordsSuspect-m-00001`.<br><br>This applies to Interflow Match jobs. |
| **Malformed Records in Input Files** | Input file records with a blank group-by column are discarded as malformed records and inserted into files with the file naming convention `malformedRecords-m-<5 digit numeral>`.<br><br>For example, `malformedRecords-m-00000`, `malformedRecords-m-00001`.<br><br>This applies to the jobs Intraflow Match, Transactional Match, Best of Breed, Duplicate Synchronization, and Filter. |

*Counters for Malformed Records*

The number of malformed records in a job run is stored in the counters:

- MALFORMED_CANDIDATE_RECORDS
- MALFORMED_SUSPECT_RECORDS
- MALFORMED_RECORDS

> **Note:** The values in these counters can be accessed by invoking the `getCounters()` method of the `AdvanceMatchFactory` instance.

## Best of Breed

Best of Breed consolidates duplicate records by selecting the best data in a duplicate record collection and creating a new consolidated record using the best data. This "super" record is known as the best of breed record. You define the rules to use in selecting records to process. When processing completes, the best of breed record is retained by the system.

## Best of Breed Job

The Best of Breed job consolidates duplicate records by selecting the best data in a duplicate record collection and creating a new consolidated record using the best data.

## API Entities

### *BestOfBreedConfiguration*

To specify the consolidation rules and the template rules to perform the Best of Breed consolidation job.

### *BestofBreedDetail*

#### *Purpose*

To specify details of a Best of Breed consolidation job.

## Input Parameters

| Parameter | Description |
|---|---|
| Group-By Option | Specify the field using which a single best of breed record is created by merging a group of similar records. A best of breed record is created for each group of records.<br>For a *MapReduce* job, pass the arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped.<br>**Number of Reducer Tasks**<br>The number of reducer tasks required to group the records.<br>For a *Spark* job, pass the arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped. |
| Best of Breed Configuration | Define the consolidation and template rules using which the best of breed record is to be created for each collection of similar records. |

| Parameter | Description |
|---|---|
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |
| Compress Output | Flag to indicate if the output must be compressed.<br>Set this to `true` to compress the output. |

## Output Columns

In addition to the input columns, the following columns are added while generating the output of a Best of Breed job:

| Parameter | Description | Output Value |
|---|---|---|
| Collection Record Type | Identifies the template and best of breed records in a collection of duplicate records. | If a template record is defined, the possible values are:<br>**Primary**<br>If the record is the selected template record in a collection.<br>**Secondary**<br>If the record is not the selected template record in a collection.<br>**BestOfBreed**<br>If the record is the newly created best of breed record in the collection.<br>If no template record is defined, the only possible value is **BestOfBreed**. |

**Note:** Other output columns, apart from **Collection Record Type**, are displayed only if they are defined while creating the consolidation conditions for the Best of Breed configuration.

### Using a Best of Breed MapReduce Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Best of Breed job by creating an instance of `BestofBreedDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.
   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

      Use an instance of **GroupbyMROption** on page 33 to specify the group-by column and the number of reducers required.

   b) Generate the consolidation and template rules for the job by creating an instance of `BestOfBreedConfiguration`. Within this instance:

      1. Define the template record for the consolidation using an instance of `ConsolidationCondition`, which comprises of `ConsolidationRule` instances.
      2. Define the consolidation conditions using instances of `ConsolidationCondition`, and connecting the conditions using logical operators.

Each instance of `ConsolidationCondition` is defined using a `ConsolidationRule` instance and its corresponding `ConsolidationAction` instance.

**Note:** Each instance of `ConsolidationRule` can be defined either using a single instance of `SimpleRule`, or using a hierarchy of child `SimpleRule` instances and nested `ConjoinedRule` instances joined using logical operators. See **Enum JoinType** on page 358 and **Enum Operation** on page 357.

c) Create an instance of `BestofBreedDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `BestOfBreedConfiguration` instance created above as the arguments to its constructor.

The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.

d) Set the details of the input file using the `inputPath` field of the `BestofBreedDetail` instance.

- For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

e) Set the details of the output file using the `outputPath` field of the `BestofBreedDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `BestofBreedDetail` instance.

g) Set the `compressOutput` flag of the `BestofBreedDetail` instance to `true` to compress the output of the job.

3. To create a MapReduce job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `createJob()`. In this, pass the above instance of `BestofBreedDetail` as an argument.

   The `createJob()` method creates the job and returns a `List` of instances of `ControlledJob`.

4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters after successful MapReduce job run, use the previously created instance of `AdvanceMatchFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using a Best of Breed Spark Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Best of Breed job by creating an instance of `BestofBreedDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.
   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

      Use an instance of **GroupbySparkOption** on page 33 to specify the group-by column.

   b) Generate the consolidation and template rules for the job by creating an instance of `BestOfBreedConfiguration`. Within this instance:

      1. Define the template record for the consolidation using an instance of `ConsolidationCondition`, which comprises of `ConsolidationRule` instances.
      2. Define the consolidation conditions using instances of `ConsolidationCondition`, and connecting the conditions using logical operators.

         Each instance of `ConsolidationCondition` is defined using a `ConsolidationRule` instance and its corresponding `ConsolidationAction` instance.

         **Note:** Each instance of `ConsolidationRule` can be defined either using a single instance of `SimpleRule`, or using a hierarchy of child `SimpleRule` instances and nested `ConjoinedRule` instances joined using logical operators. See **Enum JoinType** on page 358 and **Enum Operation** on page 357.

   c) Create an instance of `BestofBreedDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `BestOfBreedConfiguration` instance created above as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.

   d) Set the details of the input file using the `inputPath` field of the `BestofBreedDetail` instance.

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
      - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
      - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

   e) Set the details of the output file using the `outputPath` field of the `BestofBreedDetail` instance.

      - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.

- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `BestofBreedDetail` instance.

g) Set the `compressOutput` flag of the `BestofBreedDetail` instance to `true` to compress the output of the job.

3. To create and run the Spark job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `BestofBreedDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. Display the counters to view the reporting statistics for the job.

# Duplicate Synchronization

Duplicate Synchronization determines which fields from a collection of records to copy to the corresponding fields of all records in the collection. You can specify the rules that records must satisfy in order to copy the field data to the other records in the collection. When processing has been completed, all records in the collection are retained.

## Duplicate Synchronization Job

The Duplicate Synchronization job allows you to determine which fields from a collection of records to copy to the corresponding fields of all records in the collection.

## API Entities

### *DuplicateSynchronizationConfiguration*

To specify the consolidation rules to perform the Duplicate Synchronization consolidation job.

### *DuplicateSyncDetail*

*Purpose*

To specify details of a Duplicate Synchronization consolidation job.

## Input Parameters

| Parameter | Description |
|---|---|
| Group-By Option | Specifies the field to use to create groups of records to synchronize.<br>For a *MapReduce* job, pass the arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped.<br>**Number of Reducer Tasks**<br>The number of reducer tasks required to group the records.<br>For a *Spark* job, to create a Group-By option pass the arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped.<br>**Note:** If there is no group in the input, then set this parameter to null. In this case, the entire data is considered in a single group. |
| Duplicate Synchronization Configuration | The rules based on which the fields of one record are copied to the other records of a collection. |

| Parameter | Description |
|---|---|
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |
| Compress Output | Flag to indicate if the output must be compressed.<br>Set this to `true` to compress the output. |

## Output Columns

Based on the consolidation conditions defined in the *Duplicate Synchronization Configuration* input parameter, columns may be added to the output in addition to the input columns, as required.

## Using a Duplicate Synchronization MapReduce Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.

2. Provide the input and output details for the Duplicate Synchronization job by creating an instance of `DuplicateSyncDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.
   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

      Use an instance of **GroupbyMROption** on page 33 to specify the group-by column and the number of reducers required.

   b) Generate the consolidation conditions for the job by creating an instance of `DuplicateSynchronizationConfiguration`. Within this instance, define the consolidation conditions using instances of `ConsolidationCondition`, and connecting the conditions using logical operators.

      Each instance of `ConsolidationCondition` is defined using a `ConsolidationRule` instance and its corresponding `ConsolidationAction` instance.

      > **Note:** Each instance of `ConsolidationRule` can be defined either using a single instance of `SimpleRule`, or using a hierarchy of child `SimpleRule` instances and nested `ConjoinedRule` instances joined using logical operators. See **Enum JoinType** on page 358 and **Enum Operation** on page 357.

   c) Create an instance of `DuplicateSyncDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `DuplicateSynchronizationConfiguration` instance created above as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.

   d) Set the details of the input file using the `inputPath` field of the `DuplicateSyncDetail` instance.

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
      - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
      - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

   e) Set the details of the output file using the `outputPath` field of the `DuplicateSyncDetail` instance.

      - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
      - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
      - For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

   f) Set the name of the job using the `jobName` field of the `DuplicateSyncDetail` instance.

g) Set the `compressOutput` flag of the `DuplicateSyncDetail` instance to `true` to compress the output of the job.

3. Create the job by using the previously created instance of `AdvanceMatchFactory` to invoke its method `createJob()`. In this, pass the above instance of `DuplicateSyncDetail` as an argument.
   The `createJob()` method returns a `List` of instances of `ControlledJob`.
4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters after successful MapReduce job run, use the previously created instance of `AdvanceMatchFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using a Duplicate Synchronization Spark Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Duplicate Synchronization job by creating an instance of `DuplicateSyncDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.
   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

      Use an instance of **GroupbySparkOption** on page 33 to specify the group-by column.

   b) Generate the consolidation conditions for the job by creating an instance of `DuplicateSynchronizationConfiguration`. Within this instance, define the consolidation conditions using instances of `ConsolidationCondition`, and connecting the conditions using logical operators.

      Each instance of `ConsolidationCondition` is defined using a `ConsolidationRule` instance and its corresponding `ConsolidationAction` instance.

      > **Note:** Each instance of `ConsolidationRule` can be defined either using a single instance of `SimpleRule`, or using a hierarchy of child `SimpleRule` instances and nested `ConjoinedRule` instances joined using logical operators. See **Enum JoinType** on page 358 and **Enum Operation** on page 357.

   c) Create an instance of `DuplicateSyncDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `DuplicateSynchronizationConfiguration` instance created above as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.

   d) Set the details of the input file using the `inputPath` field of the `DuplicateSyncDetail` instance.

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.

- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

e) Set the details of the output file using the `outputPath` field of the `DuplicateSyncDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `DuplicateSyncDetail` instance.

g) Set the `compressOutput` flag of the `DuplicateSyncDetail` instance to `true` to compress the output of the job.

3. To create and run the Spark job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `DuplicateSyncDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. Display the counters to view the reporting statistics for the job.

# Filter

The Filter stage retains or removes records from a group of records based on the rules you specify.

## Filter Job

The Filter job retains or removes records from a group of records based on the rules you specify.

### API Entities

#### *FilterConfiguration*

To specify the consolidation rules to perform the Filter consolidation job.

#### *FilterDetail*

*Purpose*

To specify details of a Filter consolidation job.

## Input Parameters

| Parameter | Description |
|---|---|
| Group-By Option | Specifies the field to use to create groups of records to filter. The Filter job retains one or more records from each group.<br>For a *MapReduce* job, pass the arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped.<br>**Number of Reducer Tasks**<br>The number of reducer tasks required to group the records.<br>For a *Spark* job, to create a Group-By option pass the arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped.<br>**Note:** If there is no group in the input, then set this parameter to null. In this case, the entire data is considered in a single group. |
| Filter Configuration | Defines the consolidation conditions based on which the job retains one or more records from each group. |

| Parameter | Description |
| --- | --- |
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |
| Compress Output | Flag to indicate if the output must be compressed.<br>Set this to `true` to compress the output. |

## Output Columns

The output columns are the same as the input columns. No additional columns are added in the output.

## Using a Filter MapReduce Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Filter job by creating an instance of `FilterDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.

a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

   Use an instance of **GroupbyMROption** on page 33 to specify the group-by column and the number of reducers required.

b) Generate the consolidation rules for the job by creating an instance of `FilterConfiguration`. Within this instance, define the consolidation conditions using instances of `ConsolidationCondition`, and connecting the conditions using logical operators.

   Each instance of `ConsolidationCondition` is defined using a `ConsolidationRule` instance and its corresponding `ConsolidationAction` instance.

   > **Note:** Each instance of `ConsolidationRule` can be defined either using a single instance of `SimpleRule`, or using a hierarchy of child `SimpleRule` instances and nested `ConjoinedRule` instances joined using logical operators. See **Enum JoinType** on page 358 and **Enum Operation** on page 357.

c) Create an instance of `FilterDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `FilterConfiguration` instance created above as the arguments to its constructor.

   The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.

d) Set the details of the input file using the `inputPath` field of the `FilterDetail` instance.

   - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
   - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
   - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

e) Set the details of the output file using the `outputPath` field of the `FilterDetail` instance.

   - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
   - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
   - For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `FilterDetail` instance.

g) Set the `compressOutput` flag of the `FilterDetail` instance to `true` to compress the output of the job.

3. Create the job by using the previously created instance of `AdvanceMatchFactory` to invoke its method `createJob()`. In this, pass the above instance of `FilterDetail` as an argument. The `createJob()` method returns a `List` of instances of `ControlledJob`.

4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters after successful MapReduce job run, use the previously created instance of `AdvanceMatchFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using a Filter Spark Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Filter job by creating an instance of `FilterDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.
   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

      Use an instance of **GroupbySparkOption** on page 33 to specify the group-by column.

   b) Generate the consolidation rules for the job by creating an instance of `FilterConfiguration`. Within this instance, define the consolidation conditions using instances of `ConsolidationCondition`, and connecting the conditions using logical operators.

      Each instance of `ConsolidationCondition` is defined using a `ConsolidationRule` instance and its corresponding `ConsolidationAction` instance.

      > **Note:** Each instance of `ConsolidationRule` can be defined either using a single instance of `SimpleRule`, or using a hierarchy of child `SimpleRule` instances and nested `ConjoinedRule` instances joined using logical operators. See **Enum JoinType** on page 358 and **Enum Operation** on page 357.

   c) Create an instance of `FilterDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `FilterConfiguration` instance created above as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.

   d) Set the details of the input file using the `inputPath` field of the `FilterDetail` instance.

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
      - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
      - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

   e) Set the details of the output file using the `outputPath` field of the `FilterDetail` instance.

      - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.

- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `FilterDetail` instance.

g) Set the `compressOutput` flag of the `FilterDetail` instance to `true` to compress the output of the job.

3. To create and run the Spark job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `FilterDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. Display the counters to view the reporting statistics for the job.

## Interflow Match

Interflow Match locates matches between similar data records across two input record streams. The first record stream is a source for suspect records and the second stream is a source for candidate records.

Using match group criteria (for example a match key), Interflow Match identifies a group of records that are potentially duplicates of a particular suspect record.

### *Reporting*

The Interflow Match job allows you to monitor the results of the job. The counters available are:

| | |
|---|---|
| **DUPLICATE_COLLECTIONS** | The number of duplicate collections, which consist of a suspect and its duplicate records grouped together by a CollectionNumber. |
| **EXPRESS_MATCHES** | The number of Express Matches made in a collection.<br><br>An Express Match is made when a suspect and candidate have an exact match on the contents of a designated field, usually an ExpressMatchKey provided by the Match Key Generator. If an Express Match is made, no further processing is done to determine if the suspect and candidate are duplicates. |
| **AVERAGE_SCORE** | The average match score of all duplicates.<br><br>The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match. |
| **INPUT_SUSPECTS** | The number of records in the input stream that the matcher tried to match to other records. |

| | |
|---|---|
| **SUSPECTS_WITH_DUPLICATES** | The number of input suspects that matched at least one candidate record. |
| **UNIQUE_SUSPECTS** | The number of input suspects that did not match any candidate records. |
| **SUSPECTS_WITH_CANDIDATES** | The number of input suspects that had at least one candidate record in its match group and therefore had at least one match attempt. |
| **SUSPECTS_WITHOUT_CANDIDATES** | The number of input suspects that had no candidate records in its match group and therefore had no match attempts. |
| **TOTAL_DUPLICATE_CANDIDATES** | The total number of duplicate candidates found. |
| **TOTAL_DUPLICATE_SCORE** | The total match score of all the duplicates. |

## Interflow Job

The Interflow job allows you to generate a Match Key, group records using the Match Key, and perform intermatching on records from different data sources.

## API Entities

### *InterMatchDetail*

*Purpose*
To specify details of an Interflow Match job.

### *InterMatchComparisonOption*

*Purpose*
To specify comparison options while defining an Interflow Match job, whether the suspect record must be compared to all candidate records, or to any selected candidate record.

## Input Parameters

| Parameter | Description |
|---|---|
| Group-By Option | For a *MapReduce* job, pass these arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped.<br>**Number of Reducer Tasks**<br>The number of reducer tasks required to group the records.<br>For a *Spark* job, to create a Group-By option pass these arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped. |
| Match Rule | Defines as many parent and child rules as required, to create a `MatchRule` object.<br>For more information, see **MatchRule** on page 34. |

| Parameter | Description |
|---|---|
| Candidate File | *For text files:*<br>**File Path**<br>The path of the candidate text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the candidate file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the candidate file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the candidate file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the suspect file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>**Important:** The suspect and candidate files must be of the same format. Either text files or ORC format files.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
| --- | --- |
| Suspect File | *For text files:*<br>**File Path**<br>The path of the suspect text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the suspect file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the suspect file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the suspect file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the suspect file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |
| Match Key Settings | A combination of the columns and the algorithms to be applied to generate the match key, required to perform the matching.<br>**Note:** Specify only one match key.<br>**Attention:** Set the match key settings only if you wish to generate a match key before performing the matching. |
| Job Name | The name of the job. |
| Express Match Column | The name of the column to be used for express matching of records. |

| Parameter | Description |
|---|---|
| Setting Collection Number Zero to Unique Records | Set this to `true` to set the collection number of unique records as 0 (zero). |
| Comparison Option | Allows you to select one of the two options:<br>• Compare the Suspect record to all Candidate records: Specify whether unique records must be returned in the output or not.<br>• Compare the Suspect record to the selected Candidate record only: Specify the maximum number of duplicate records to be searched and returned. |
| Compress Output | Flag to indicate if the output must be compressed.<br>Set this to `true` to compress the output. |

## Output Columns

In addition to the input columns, the following columns are added while generating the output of an Interflow Match job:

| Column | Description | Output Value |
|---|---|---|
| Collection Number | Identifies a collection of duplicate records. | The possible values are `0-0-1`, `0-0-2`, and the like. |
| Express Match Identified | Indicates whether the match was obtained using the express match key. | 1. For a duplicate candidate record matched using an express match key, the output value is `Y`.<br>2. For a duplicate candidate record matched, but not using an express match key, the output value is blank.<br>3. For a unique candidate record matched using an express match key, the output value is `N`.<br>4. For a suspect record matched using an express match key, the output value is blank. |
| Interflow Source Type | Indicates whether the input record is a suspect record or a candidate record. | The possible values are `S` for a suspect record, and `C` for a candidate record. |
| Match Record Type | Identifies the type of match record in a collection. | The possible values are `S` (suspect record), `D` (duplicate record) and `U` (unique record). |

| Column | Description | Output Value |
|---|---|---|
| Match Score | Identifies the overall score between two records. | The possible values range from 0 (zero) to 100 for duplicate and unique records, where 0 indicates a poor match and 100 indicates a very high-quality match.<br><br>**Note:** For suspect records, this value is 0. |

### Using an Interflow Match MapReduce Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Interflow Match job by creating an instance of `InterMatchDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.
   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

      Use an instance of **GroupbyMROption** on page 33 to specify the group-by column and the number of reducers required.
   b) Generate the matching rules for the job by creating an instance of `MatchRule`.
   c) Create an instance of `InterMatchDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `MatchRule` instance created above as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.
   d) Set the details of the candidate file using the `candidateFilePath` field of the `InterMatchDetail` instance.

      For a text candidate file, create an instance of `FilePath` with the relevant details of the candidate file by invoking the appropriate constructor. For an ORC candidate file, create an instance of `OrcFilePath` with the path of the ORC candidate file as the argument.
   e) Set the details of the suspect file using the `suspectFilePath` field of the `InterMatchDetail` instance.

      For a text suspect file, create an instance of `FilePath` with the relevant details of the suspect file by invoking the appropriate constructor. For an ORC suspect file, create an instance of `OrcFilePath` with the path of the ORC suspect file as the argument. For a parquet suspect file, create an instance of `ParquetFilePath` with the path of the parquet suspect file as the argument.

      **Important:** The suspect and candidate files must be of the same format. Either text files or ORC format files.
   f) Set the details of the output file using the `outputPath` field of the `InterMatchDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

g) Set the name of the job using the `jobName` field of the `InterMatchDetail` instance.

h) Set the Express Match Column using the `expressMatchColumn` field of the `InterMatchDetail` instance, if required.

i) Set the flag `collectionNumberZerotoUniqueRecords` of the `InterMatchDetail` instance to `true` to allocate the collection number 0 (zero) to a unique record. The default is `true`.

If you do not wish to allocate the collection number zero to unique records, set this flag to `false`.

j) Set the comparison option using the `comparisonOption` field of the `InterMatchDetail` instance. In this field, set the required value using the class **InterMatchComparisonOption** on page 55 to select one of the two options:

- **Compare the Suspect record to all Candidate records**: Specify whether unique records must be returned in the output or not.
- **Compare the Suspect record to the selected Candidate record only**: Specify the maximum number of duplicate records to be searched and returned.

k) Set the `compressOutput` flag of the `InterMatchDetail` instance to `true` to compress the output of the job.

l) If the input data does not have match keys, you must specify the match key settings to first run the Match Key Generator job to generate the match keys, before running the Interflow Match job.

To generate the match keys for the input data, specify the match key settings by creating and configuring an instance of `MatchKeySettings` to generate a match key before performing the interflow matching. Set this instance using the `matchKeySettings` field of the `InterMatchDetail` instance.

**Note:** To see how to set match key settings, see the code samples.

3. To create a MapReduce job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `createJob()`. In this, pass the above instance of `InterMatchDetail` as an argument.

   The `createJob()` method creates the job and returns a `List` of instances of `ControlledJob`.

4. Run the created job using an instance of `JobControl`.

5. To display the reporting counters after successful MapReduce job run, use the previously created instance of `AdvanceMatchFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using an Interflow Match Spark Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Interflow Match job by creating an instance of `InterMatchDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.
   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

      Use an instance of **GroupbySparkOption** on page 33 to specify the group-by column.

   b) Generate the matching rules for the job by creating an instance of `MatchRule`.
   c) Create an instance of `InterMatchDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `MatchRule` instance created above as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.

   d) Set the details of the candidate file using the `candidateFilePath` field of the `InterMatchDetail` instance.

      For a text candidate file, create an instance of `FilePath` with the relevant details of the candidate file by invoking the appropriate constructor. For an ORC candidate file, create an instance of `OrcFilePath` with the path of the ORC candidate file as the argument.

   e) Set the details of the suspect file using the `suspectFilePath` field of the `InterMatchDetail` instance.

      For a text suspect file, create an instance of `FilePath` with the relevant details of the suspect file by invoking the appropriate constructor. For an ORC suspect file, create an instance of `OrcFilePath` with the path of the ORC suspect file as the argument. For a parquet suspect file, create an instance of `ParquetFilePath` with the path of the parquet suspect file as the argument.

      **Important:** The suspect and candidate files must be of the same format. Either text files or ORC format files.

   f) Set the details of the output file using the `outputPath` field of the `InterMatchDetail` instance.

      - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
      - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
      - For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

g) Set the name of the job using the `jobName` field of the `InterMatchDetail` instance.

h) Set the Express Match Column using the `expressMatchColumn` field of the `InterMatchDetail` instance, if required.

i) Set the flag `collectionNumberZerotoUniqueRecords` of the `InterMatchDetail` instance to `true` to allocate the collection number 0 (zero) to a unique record. The default is `true`.

   If you do not wish to allocate the collection number zero to unique records, set this flag to `false`.

j) Set the comparison option using the `comparisonOption` field of the `InterMatchDetail` instance. In this field, set the required value using the class **InterMatchComparisonOption** on page 55 to select one of the two options:

   - **Compare the Suspect record to all Candidate records**: Specify whether unique records must be returned in the output or not.
   - **Compare the Suspect record to the selected Candidate record only**: Specify the maximum number of duplicate records to be searched and returned.

k) Set the `compressOutput` flag of the `InterMatchDetail` instance to `true` to compress the output of the job.

l) If the input data does not have match keys, you must specify the match key settings to first run the Match Key Generator job to generate the match keys, before running the Interflow Match job.

   To generate the match keys for the input data, specify the match key settings by creating and configuring an instance of `MatchKeySettings` to generate a match key before performing the interflow matching. Set this instance using the `matchKeySettings` field of the `InterMatchDetail` instance.

   **Note:** To see how to set match key settings, see the code samples.

3. To create and run the Spark job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `InterMatchDetail` as an argument.
   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.
4. Display the counters to view the reporting statistics for the job.

## Intraflow Match

Intraflow Match locates matches between similar data records within a single input stream. You can create hierarchical rules based on any fields that have been defined or created in other stages of the dataflow.

*Reporting*

The Intraflow Match job allows you to monitor the results of the job. The counters available are:

| | |
|---|---|
| **INPUT_RECORDS** | The number of records in the matching stage before the matching sort is performed. |
| **DUPLICATE_RECORDS** | The number of duplicate records within a match group, which can be either a suspect or a candidate record. |
| **UNIQUE_RECORDS** | The number of suspect or candidate records which do not match any other records in their respective match group.<br><br>If it is the only record in a match group, a suspect is automatically unique. |
| **MATCH_GROUPS** | (Group By) Records grouped together by a match key. |
| **DUPLICATE_COLLECTIONS** | The number of duplicate collections, which consist of a suspect and its duplicate records grouped together by a CollectionNumber. |
| **EXPRESS_MATCHES** | The number of Express Matches made in a collection.<br><br>An Express Match is made when a suspect and candidate have an exact match on the contents of a designated field, usually an ExpressMatchKey provided by the Match Key Generator. If an Express Match is made, no further processing is done to determine if the suspect and candidate are duplicates. |
| **AVERAGE_SCORE** | The average match score of all duplicates.<br><br>The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match. |
| **TOTAL_DUPLICATES** | The total number of duplicates found. |
| **TOTAL_SCORE** | The total match score of all duplicates. |

## Intraflow Job

The Intraflow job allows you to generate a Match Key, group records using the Match Key, and perform intramatching on records from the same data source.

## API Entities

### *IntraMatchDetail*

*Purpose*

To specify details of an Intraflow Match job.

## Input Parameters

| Parameter | Description |
|---|---|
| Group-By Option | For a *MapReduce* job, pass these arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped.<br>**Number of Reducer Tasks**<br>The number of reducer tasks required to group the records.<br>For a *Spark* job, to create a Group-By option pass these arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped. |
| Match Rule | Defines as many parent and child rules as required, to create a `MatchRule` object.<br>For more information, see **MatchRule** on page 34. |

| Parameter | Description |
| --- | --- |
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |
| Job Name | The name of the job. |
| Express Match Column | The name of the column to be used for express matching of records. |
| Setting Collection Number Zero to Unique Records | Set this to `true` to set the collection number of unique records as 0 (zero). |
| Compress Output | Flag to indicate if the output must be compressed.<br>Set this to `true` to compress the output. |

| Parameter | Description |
|---|---|
| Match Key Settings | A combination of the columns and the algorithms to be applied to generate the match key, required to perform the matching.<br>**Note:** Specify only one match key.<br>**Attention:** Set the match key settings only if you wish to generate a match key before performing the matching. |

## Output Columns

In addition to the input columns, the following columns are added while generating the output of an Intraflow Match job:

| Column | Description | Output Value |
|---|---|---|
| Collection Number | Identifies a collection of duplicate records. | The possible values are `0-0-1`, `0-0-2`, and the like. |
| Express Match Identified | Indicates whether the match was obtained using the express match key. | 1. For a duplicate candidate record matched using an express match key, the output value is `Y`.<br>2. For a duplicate candidate record matched, but not using an express match key, the output value is blank.<br>3. For a unique candidate record matched using an express match key, the output value is blank.<br>4. For a suspect record matched using an express match key, the output value is blank. |
| Match Record Type | Identifies the type of match record in a collection. | The possible values are `S` (suspect record), `D` (duplicate record) and `U` (unique record). |
| Match Score | Identifies the overall score between two records. | The possible values range from 0 (zero) to 100 for duplicate and unique records, where 0 indicates a poor match and 100 indicates a very high-quality match.<br>**Note:** For suspect records, this value is 0. |

### Using an Intraflow Match MapReduce Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Intraflow Match job by creating an instance of `IntraMatchDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.
   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

      Use an instance of **GroupbyMROption** on page 33 to specify the group-by column and the number of reducers required.
   b) Generate the matching rules for the job by creating an instance of `MatchRule`.
   c) Create an instance of `IntraMatchDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `MatchRule` instance created above as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.
   d) Set the details of the input file using the `inputPath` field of the `IntraMatchDetail` instance.

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
      - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
      - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

   e) Set the details of the output file using the `outputPath` field of the `IntraMatchDetail` instance.

      - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
      - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
      - For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

   f) Set the name of the job using the `jobName` field of the `IntraMatchDetail` instance.
   g) Set the Express Match Column using the `expressMatchColumn` field of the `IntraMatchDetail` instance, if required.
   h) Set the flag `collectionNumberZerotoUniqueRecords` of the `IntraMatchDetail` instance to `true` to allocate the collection number 0 (zero) to a unique record. The default is `true`.

      If you do not wish to allocate the collection number zero to unique records, set this flag to `false`.

i) Set the `compressOutput` flag of the `IntraMatchDetail` instance to `true` to compress the output of the job.

j) If the input data does not have match keys, you must specify the match key settings to first run the Match Key Generator job to generate the match keys, before running the Intraflow Match job.

To generate the match keys for the input data, specify the match key settings by creating and configuring an instance of `MatchKeySettings` to generate a match key before performing the intraflow matching. Set this instance using the `matchKeySettings` field of the `IntraMatchDetail` instance.

**Note:** To see how to set match key settings, see the code samples.

3. To create a MapReduce job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `createJob()`. In this, pass the above instance of `IntraMatchDetail` as an argument.

   The `createJob()` method creates the job and returns a `List` of instances of `ControlledJob`.

4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters after successful MapReduce job run, use the previously created instance of `AdvanceMatchFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using an Intraflow Match Spark Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Intraflow Match job by creating an instance of `IntraMatchDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.

   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

   Use an instance of **GroupbySparkOption** on page 33 to specify the group-by column.

   b) Generate the matching rules for the job by creating an instance of `MatchRule`.

   c) Create an instance of `IntraMatchDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `MatchRule` instance created above as the arguments to its constructor.

   The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.

   d) Set the details of the input file using the `inputPath` field of the `IntraMatchDetail` instance.

   - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
   - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.

- For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

e) Set the details of the output file using the `outputPath` field of the `IntraMatchDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `IntraMatchDetail` instance.

g) Set the Express Match Column using the `expressMatchColumn` field of the `IntraMatchDetail` instance, if required.

h) Set the flag `collectionNumberZerotoUniqueRecords` of the `IntraMatchDetail` instance to `true` to allocate the collection number 0 (zero) to a unique record. The default is `true`.

If you do not wish to allocate the collection number zero to unique records, set this flag to `false`.

i) Set the `compressOutput` flag of the `IntraMatchDetail` instance to `true` to compress the output of the job.

j) If the input data does not have match keys, you must specify the match key settings to first run the Match Key Generator job to generate the match keys, before running the Intraflow Match job.

To generate the match keys for the input data, specify the match key settings by creating and configuring an instance of `MatchKeySettings` to generate a match key before performing the intraflow matching. Set this instance using the `matchKeySettings` field of the `IntraMatchDetail` instance.

**Note:** To see how to set match key settings, see the code samples.

3. To create and run the Spark job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `IntraMatchDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. Display the counters to view the reporting statistics for the job.

# Match Key Generator

Match Key Generator creates a non-unique key for each record, which can then be used by matching stages to identify groups of potentially duplicate records. Match keys facilitate the matching process by allowing you to group records by match key and then only comparing records within these groups.

The match key is created using rules you define and is comprised of input fields. Each input field specified has a selected algorithm that is performed on it. The result of each algorithm is then concatenated to create a single match key field.

In addition to creating match keys, you can also create express match keys to be used later in the dataflow by an Intraflow Match stage or an Interflow Match stage.

You can create multiple match keys and express match keys.

For example, if the incoming record is:

First Name - Fred
Last Name - Mertz
Postal Code - 21114-1687
Gender Code - M

And you define a match key rule that generates a match key by combining data from the record like this:

| Input Field | Start Position | Length |
|---|---|---|
| Postal Code | 1 | 5 |
| Postal Code | 7 | 4 |
| Last Name | 1 | 5 |
| First Name | 1 | 5 |
| Gender Code | 1 | 1 |

Then the key would be:

211141687MertzFredM

## Match Key Generator job

The Match Key Generator job allows you to generate Match Keys.

**Note:** To generate a match key for the data, you must run the Match Key Generator job once before running any other jobs.

## API Entities

### *MatchKeyGeneratorDetail*

*Purpose*
To specify details of a Match Key Generator job.

## Input Parameters

| Parameter | Description |
|---|---|
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |
| Match Key Settings | A combination of the columns and the algorithms to be applied to generate the match key, required to perform the matching.<br>**Note:** At least one match key must be specified. You can specify more than one match keys, if required. |
| Job Name | The name of the job. |

## Output Columns

In addition to the input columns, these columns are added while generating the output of a Match Key Generator job:

| Column | Description | Output Value |
|---|---|---|
| MatchKey | The key generated to identify records. | The key generated depending on the columns and algorithms selected to generate the match key.<br><br>**Note:** The number of user-named match key columns generated in the output depends on the job settings. |

## Using a Match Key Generator MapReduce Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Match Key Generator job by creating an instance of `MatchKeyGeneratorDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.
   a) Specify the match key settings to perform the matching by creating and configuring an instance of `MatchKeySettings`. For more information, see the relevant code sample.
   b) Create an instance of `MatchKeyGeneratorDetail` by passing an instance of type `JobConfig` and the `MatchKeySettings` instance created as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.
   c) Set the details of the input file using the `inputPath` field of the `MatchKeyGeneratorDetail` instance.

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
      - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
      - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

   d) Set the details of the output file using the `outputPath` field of the `MatchKeyGeneratorDetail` instance.

      - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
      - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
      - For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

   e) Set the name of the job using the `jobName` field of the `MatchKeyGeneratorDetail` instance.

3. To create a MapReduce job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `createJob()`. In this, pass the above instance of `MatchKeyGeneratorDetail` as an argument.

   The `createJob()` method creates the job and returns a `List` of instances of `ControlledJob`.

4. Run the created job using an instance of `JobControl`.

### Using a Match Key Generator Spark Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide input and output details for the Match Key Generator job by creating an instance of `MatchKeyGeneratorDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.
   a) Specify the match key settings to perform the matching by creating and configuring an instance of `MatchKeySettings`. For more information, see the relevant code sample.
   b) Create an instance of `MatchKeyGeneratorDetail` by passing an instance of type `JobConfig` and the `MatchKeySettings` instance created as arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.

   c) Set the details of the input file using the `inputPath` field of the `MatchKeyGeneratorDetail` instance.

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
      - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
      - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

   d) Set the details of the output file using the `outputPath` field of the `MatchKeyGeneratorDetail` instance.

      - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
      - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
      - For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

   e) Set the name of the job using the `jobName` field of the `MatchKeyGeneratorDetail` instance.
3. To create and run the Spark job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `MatchKeyGeneratorDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

# Transactional Match

Transactional Match matches suspect records against candidate records of a group of records to identify duplicates. The records are first grouped by a selected column, post which the first record is marked as the suspect record. All the remaining records of the group, termed as candidate records, are matched against the suspect record.

If the candidate record is a duplicate, it is assigned a collection number, the match record type is labeled a Duplicate, and the record is then written out. Any unmatched candidates in the group are assigned a collection number of 0, labeled as Unique and then written out as well.

*Reporting*

The Transactional Match job allows you to monitor the results of the job. The counters available are:

| | |
|---|---|
| **AVERAGE_SCORE** | The average match score of all duplicates.<br>The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match. |
| **INPUT_SUSPECTS** | The number of records in the input stream that the matcher tried to match to other records. |
| **SUSPECTS_WITH_DUPLICATES** | The number of input suspects that matched at least one candidate record. |
| **UNIQUE_SUSPECTS** | The number of input suspects that did not match any candidate records. |
| **SUSPECTS_WITH_CANDIDATES** | The number of input suspects that had at least one candidate record in its match group and therefore had at least one match attempt. |
| **SUSPECTS_WITHOUT_CANDIDATES** | The number of input suspects that had no candidate records in its match group and therefore had no match attempts. |
| **TOTAL_DUPLICATES_SCORE** | The total match score of all duplicates. |
| **TOTAL_DUPLICATES** | The total number of duplicates found. |

## Transactional Match Job

The Transactional Match job allows you to match suspect records against candidate records of a group of records to identify duplicates.

### API Entities

#### *TransactionalMatchDetail*

*Purpose*

To specify details of a Transactional Match job.

## Input Parameters

| Parameter | Description |
|---|---|
| Group-By Option | For a *MapReduce* job, pass these arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped.<br>**Number of Reducer Tasks**<br>The number of reducer tasks required to group the records.<br>For a *Spark* job, to create a Group-By option pass these arguments:<br>**GroupBy Column**<br>The name of the column using which the records are to be grouped. |
| Match Rule | Defines as many parent and child rules as required, to create a `MatchRule` object.<br>For more information, see **MatchRule** on page 34. |

| Parameter | Description |
| --- | --- |
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |
| Return Unique Candidates | Flag to indicate whether unique candidates must be returned as part of the output. |
| Compress Output | Flag to indicate if the output must be compressed.<br>Set this to `true` to compress the output. |

| Parameter | Description |
|---|---|
| Match Key Settings | A combination of the columns and the algorithms to be applied to generate the match key, required to perform the matching.<br>**Note:** Specify only one match key.<br>**Attention:** Set the match key settings only if you wish to generate a match key before performing the matching. |

## Output Columns

In addition to the input columns, the following columns are added while generating the output of a Transactional Match job:

| Parameter | Description | Output Value |
|---|---|---|
| Match Record Type | Identifies the type of match record in a collection. | The possible values are `S` (suspect record), `D` (duplicate record) and `U` (unique record). |
| Match Score | Identifies the overall score between two records. | The possible values range from 0 (zero) to 100 for duplicate and unique records, where 0 indicates a poor match and 100 indicates a very high-quality match.<br>**Note:** For suspect records, this value is 0. |
| Has Duplicates | Indicates whether the suspect records has duplicates or not | For Suspect records, the possible output values are:<br>• `Y` (if duplicates are present)<br>• `N` (if duplicates are absent)<br>For Duplicate records, the output value is `D`.<br>For Unique records, the output value is `U`. |

## Using a Transactional Match MapReduce Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Transactional Match job by creating an instance of `TransactionalMatchDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.
   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

Use an instance of **GroupbyMROption** on page 33 to specify the group-by column and the number of reducers required.

b) Generate the matching rules for the job by creating an instance of `MatchRule`.

c) Create an instance of `TransactionalMatchDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `MatchRule` instance created above as the arguments to its constructor.

The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.

d) Set the details of the input file using the `inputPath` field of the `TransactionalMatchDetail` instance.

- For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

e) Set the details of the output file using the `outputPath` field of the `TransactionalMatchDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `TransactionalMatchDetail` instance.

g) Set the flag `returnUniqueCandidates` of the `TransactionalMatchDetail` instance to `true` to return unique candidate records in the output. The default is `true`.

h) Set the `compressOutput` flag of the `TransactionalMatchDetail` instance to `true` to compress the output of the job.

i) If the input data does not have match keys, you must specify the match key settings to first run the Match Key Generator job to generate the match keys, before running the Transactional Match job.

To generate the match keys for the input data, specify the match key settings by creating and configuring an instance of `MatchKeySettings` to generate a match key before performing the transactional matching. Set this instance using the `matchKeySettings` field of the `TransactionalMatchDetail` instance.

**Note:** To see how to set match key settings, see the code samples.

3. To create a MapReduce job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `createJob()`. In this, pass the above instance of `TransactionalMatchDetail` as an argument.

   The `createJob()` method creates the job and returns a `List` of instances of `ControlledJob`.
4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters after successful MapReduce job run, use the previously created instance of `AdvanceMatchFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using a Transactional Match Spark Job

1. Create an instance of `AdvanceMatchFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Transactional Match job by creating an instance of `TransactionalMatchDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.
   a) Specify the column using which the records are to be grouped by creating an instance of `GroupbyOption`.

      Use an instance of **GroupbySparkOption** on page 33 to specify the group-by column.
   b) Generate the matching rules for the job by creating an instance of `MatchRule`.
   c) Create an instance of `TransactionalMatchDetail`, by passing an instance of type `JobConfig`, the `GroupbyOption` instance created, and the `MatchRule` instance created above as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.
   d) Set the details of the input file using the `inputPath` field of the `TransactionalMatchDetail` instance.

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
      - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
      - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

   e) Set the details of the output file using the `outputPath` field of the `TransactionalMatchDetail` instance.

      - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
      - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
      - For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `TransactionalMatchDetail` instance.

g) Set the flag `returnUniqueCandidates` of the `TransactionalMatchDetail` instance to `true` to return unique candidate records in the output. The default is `true`.

h) Set the `compressOutput` flag of the `TransactionalMatchDetail` instance to `true` to compress the output of the job.

i) If the input data does not have match keys, you must specify the match key settings to first run the Match Key Generator job to generate the match keys, before running the Transactional Match job.

To generate the match keys for the input data, specify the match key settings by creating and configuring an instance of `MatchKeySettings` to generate a match key before performing the transactional matching. Set this instance using the `matchKeySettings` field of the `TransactionalMatchDetail` instance.

**Note:** To see how to set match key settings, see the code samples.

3. To create and run the Spark job, use the previously created instance of `AdvanceMatchFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `TransactionalMatchDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. Display the counters to view the reporting statistics for the job.

# Data Integration Module Jobs

## Common Module API

### DataIntegrationFactory

*Purpose*
A singleton factory class to create instances of **Data Integration Module** jobs.

# Custom Groovy Script

## Custom Groovy Script Job

This job transforms the input fields based on the defined groovy script.

## API Entities

### *CustomGroovyScriptConfiguration*

To specify these details:

- Groovy script file
- Input fields
- Output fields

### *CustomGroovyScriptDetail*

#### *Purpose*

To specify these for a `Custom Groovy Script Detail` job:

- Input file
- Output file
- Name of the job
- Date pattern as `M/d/yy`
- Date-time pattern as `M/d/yy h:mm a`
- Time pattern as `h:mm a`

## Input Parameters

| Parameter | Description |
|---|---|
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30.<br>For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |

| Parameter | Description |
|---|---|
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |

## Output Columns

The output columns are the same as the input columns, with the values transformed through the Groovy script.

## Using a Groovy Script MapReduce Job

1. Create an instance of `DataIntegrationFactory` by using its static method `getInstance()`.
2. Provide the input and output details for the GroovyScript job by creating an instance of `CustomGroovyScriptDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32. Use these steps to create and configure the `CustomGroovyScriptDetail` instance.

a) Create an instance of `CustomGroovyScriptDetail` by specifying the `ProcessType` as **MRProcessType** on page 32. To this instance, set these details:

- Input file: Use the `inputPath` field

  **Note:**

  - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
  - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
  - For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

- Output file: Use the `outputPath` field

  **Note:**

  - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
  - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
  - For a parquet output file, create an instance of `ParquetFilePath` with the path of the parquet output file as the argument.

- Name of the job: Use the `jobName` field
- Date pattern: `M/d/yy`
- Date-time pattern: `M/d/yy h:mm a`
- Time pattern: `h:mm a`

b) Create an instance of `CustomGroovyScriptConfiguration` and set these details to it:

- The groovyScriptFile
- InputFields
- OutputFields

c) Create a `configuration` by using the `getScriptTransformerConfiguration()` method, which calls the list of `CustomGroovyScriptConfiguration` instances created and configured above.

3. To create a MapReduce job, use the previously created instance of `DataIntegrationFactory` to invoke its method `createJob()`. In this, pass the above instance of `CustomGroovyScriptDetail` as an argument.

   The `createJob()` method creates the job and returns a `List` of instances of `ControlledJob`.

4. Run the created job using an instance of `JobControl`.

### Using a Groovy Script Spark Job

1. Create an instance of `DataIntegrationFactory` by using its static method `getInstance()`.
2. Provide the input and output details for the GroovyScript job by creating an instance of `CustomGroovyScriptDetail` specifying the `ProcessType`. The instance must use the **SparkProcessType** on page 32. Use these steps to create and configure the `CustomGroovyScriptDetail` instance.
   a) Create an instance of `CustomGroovyScriptDetail` by specifying the `ProcessType` as **SparkProcessType** on page 32. To this instance, set these details:
      - Input file: Use the `inputPath` field

        **Note:**

        - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
        - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
        - For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

      - Output file: Use the `outputPath` field

        **Note:**

        - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
        - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
        - For a parquet output file, create an instance of `ParquetFilePath` with the path of the parquet output file as the argument.

      - Name of the job: Use the `jobName` field
      - Date pattern: `M/d/yy`
      - Date-time pattern: `M/d/yy h:mm a`
      - Time pattern: `h:mm a`

   b) Create an instance of `CustomGroovyScriptConfiguration` and set these details to it:
      - The groovyScriptFile
      - InputFields
      - OutputFields

   c) Create a `configuration` by using the `getScriptTransformerConfiguration()` method, which calls the list of `CustomGroovyScriptConfiguration` instances created and configured above.

3. To create a Spark job, use the previously created instance of `DataIntegrationFactory` to invoke its method `runSparkJob()`. In this, pass the `JoinDetail` instance as an argument.

   The `runSparkJob()` method creates the job and returns a `map` of instances of `ControlledJob`.

# Joiner

### Joiner Job

This job performs a SQL-style JOIN operation to combine records from multiple files.

### API Entities

#### *JoinDetail*

##### *Purpose*

This class specifies details of a join job, such as input path of the files, type of Join (Inner, LeftOuter or Full), columns to be joined, and output path of the job.

### Input Parameters

| Parameter | Description |
|---|---|
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30.<br>For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |

| Parameter | Description |
|---|---|
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |

### Output Columns

The joined columns are displayed in the output columns.

### Using a Joiner MapReduce Job

1. Create an instance of `DataIntegrationFactory` by using its static method `getInstance()`.
2. Provide the input and output details for the job in the `JoinDetail` instance, specifying the `ProcessType` as **MRProcessType** on page 32. Use these steps to create and configure the `JoinDetail` instance.
    1. Create an instance of `JoinDetail` by specifying the `ProcessType` as **MRProcessType** on page 32 and using the default configurations.

2. Create separate instances of `FilePath` and for each of those, configure these input file details: `RecordSeparator` (use **Enum RecordSeparator** on page 357), `fieldSeperator`, `textQualifier`, and `fileHeader` (specify if the first row is to be skipped).

   **Note:**

   - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
   - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
   - For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

3. In the `JoinDetail` instance created in the above step, configure these details:

   - `InputPaths`: Pass the `FilePath` instances created and configured above
   - `LeftInput`: Specify the left input for the join operation
   - `JobName`: Name of the job
   - `JoinType`: Use **Enum JoinDetail.JoinType** on page 358 to define the join type
   - `JoinColumns`: Specify the input columns to be joined. These should be comma separated values.
   - `OutputPath`: Use the `setOutputPath` method to set the output path of the job, specifying if the file is to be overwritten, and header is to be created.

3. To create a `MapReduce` job, use the previously created instance of `DataIntegrationFactory` to invoke its method `createJob()`. In this, pass the `JoinDetail` instance as an argument.

   The `createJob()` method creates the job and returns a `List` of instances of `ControlledJob`.

4. Run the created job using an instance of `JobControl`.

### Using a Joiner Spark Job

1. Create an instance of `DataIntegrationFactory` by using its static method `getInstance()`.
2. Provide the input and output details for the job in the `JoinDetail` instance, specifying the `ProcessType` as **SparkProcessType** on page 32. Use these steps to create and configure the `JoinDetail` instance.

   1. Create an instance of JoinDetail by specifying the ProcessType as **SparkProcessType** on page 32 and using the default configurations.
   2. Create separate instances of `FilePath` and for each of those, configure these input file details: `RecordSeparator` (use **Enum RecordSeparator** on page 357), `fieldSeperator`, `textQualifier`, and `fileHeader` (specify if the first row is to be skipped).

      **Note:**

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.

- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

3. In the `JoinDetail` instance created in the above step, configure these details:
   - `InputPaths`: Pass the `FilePath` instances created and configured above
   - `LeftInput`: Specify the left input for the join operation
   - `JobName`: Name of the job
   - `JoinType`: Use **Enum JoinDetail.JoinType** on page 358 to define the join type
   - `JoinColumns`: Specify the input columns to be joined. These should be comma separated values.
   - `OutputPath`: Use the `setOutputPath` method to set the output path of the job, specifying if the file is to be overwritten, and header is to be created.

3. To create a Spark job, use the previously created instance of `DataIntegrationFactory` to invoke its method `runSparkJob()`. In this, pass the `JoinDetail` instance as an argument.
   The `runSparkJob()` method creates the job and returns a `map` of instances of `ControlledJob`.

# Data Normalization Module Jobs

## Common Module API

### DataNormalizationDetail<T extends ProcessType>

*Purpose*
To specify the details of a Data Normalization Module job.

### DataNormalizationFactory

*Purpose*
A singleton factory class to create instances of Data Normalization Module jobs.

# Advanced Transformer

The Advanced Transformer job scans and splits strings of data into multiple fields using tables or regular expressions. It extracts a specific term or a specified number of words to the right or left of a term. Extracted and non-extracted data can be placed into an existing field or a new field.

For example, want to extract the suite information from this address field and place it in a separate field.

2300 BIRCH RD STE 100

To accomplish this, you could create an Advanced Transformer that extracts the term STE and all words to the right of the term STE, leaving the field as:

2300 BIRCH RD

## Advanced Transformer Job

The Advanced Transformer job scans and splits strings of data into multiple fields using tables or regular expressions. It extracts a specific term or a specified number of words to the right or left of a term.

## API Entities

### *AbstractAdvancedTransformerRules*

*Purpose*
Parent class to specify the rules for an Advanced Transformer job.

### *AdvancedTransformerDetail*

*Purpose*
To specify details of an Advanced Transformer job.

### *AdvancedTransformerConfiguration*

*Purpose*
To scan and split strings of data into multiple fields using tables or regular expressions.

### *RegularExpressionExtraction*

*Purpose*
To specify rules to extract data using regular expressions.

### *RegularExpressionGroupItem*

*Purpose*
To specify a part of a parent regular expression. Each part of a parent regular expression can be stored in a different output field.

### *TableDataExtraction*

*Purpose*

To defines rules for extracting data from table.

## Input Parameters

| Parameter | Description |
|---|---|
| Advanced Transformer Configuration | To scan and split strings of data into multiple fields using tables or regular expressions.<br>Allows extraction of a specific term or a specified number of words to the right or left of a term. Extracted and non-extracted data are placed into an existing field or a new field.<br>The Advanced Transformer rules can be defined using an instance of type `AdvancedTransformerConfiguration`. This instance must be an instance of either `TableDataExtraction` or `RegularExpressionExtraction`. |
| Reference Data Path | To specify the Reference Data path details. |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |

| Parameter | Description |
| --- | --- |
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |

## Output Columns

In addition to the input columns, the following columns are added while generating the output of an Advanced Transformer job:

| Column | Description | Output Value |
|---|---|---|
| Non-Extracted Data | This output column is added if a new column name, not present in the input, is specified as the Non-Extracted Data column.<br>The name of the column is as entered by you.<br>**Note:** For the Non-Extracted Data column, you can select an existing source column or type a new column name. | The non-extracted data for the respective record based on the specified term. |
| Extracted Data | This output column is added if a new column name, not present in the input, is specified as the Extracted Data column.<br>The name of the column is as entered by you.<br>**Note:** For the Extracted Data column, you can select an existing source column or type a new column name. | The extracted data for the respective record based on the specified term. |
| Advanced Transform Term Identified | Indicates whether the term has been identified or not. | The possible value is `Yes` or `No`. |

## Using an Advanced Transformer MapReduce Job

1. Create an instance of `DataNormalizationFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Advanced Transformer job by creating an instance of `AdvancedTransformerDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.
   a) Configure the advanced transformer rules by creating an instance of `AdvancedTransformerConfiguration`.
   Within this instance, add an instance of type `AbstractAdvancedTransformerRules`. This `AbstractAdvancedTransformerRules` instance must be defined using one of these classes: `TableDataExtraction` or `RegularExpressionExtraction`, corresponding to the desired advanced transformer rule category.
   b) Set the details of the Reference Data path and location type by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.

c) Create an instance of `AdvancedTransformerDetail`, by passing an instance of type `JobConfig`, and the `AdvancedTransformerConfiguration` and `ReferenceDataPath` instances created earlier as the arguments to its constructor.

The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.

d) Set the details of the input file using the `inputPath` field of the `AdvancedTransformerDetail` instance.

- For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

e) Set the details of the output file using the `outputPath` field of the `AdvancedTransformerDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `AdvancedTransformerDetail` instance.

3. To create a MapReduce job, use the previously created instance of `DataNormalizationFactory` to invoke its method `createJob()`. In this, pass the above instance of `AdvancedTransformerDetail` as an argument.
   The `createJob()` method returns a `List` of instances of `ControlledJob`.
4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters post a successful MapReduce job run, use the previously created instance of `DataNormalizationFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using an Advanced Transformer Spark Job

1. Create an instance of `DataNormalizationFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Advanced Transformer job by creating an instance of `AdvancedTransformerDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.
   a) Configure the advanced transformer rules by creating an instance of `AdvancedTransformerConfiguration`.

Within this instance, add an instance of type `AbstractAdvancedTransformerRules`. This `AbstractAdvancedTransformerRules` instance must be defined using one of these classes: `TableDataExtraction` or `RegularExpressionExtraction`, corresponding to the desired advanced transformer rule category.

b) Set the details of the Reference Data path and location type by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.

c) Create an instance of `AdvancedTransformerDetail`, by passing an instance of type `JobConfig`, and the `AdvancedTransformerConfiguration` and `ReferenceDataPath` instances created earlier as the arguments to its constructor.

The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.

d) Set the details of the input file using the `inputPath` field of the `AdvancedTransformerDetail` instance.

- For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

e) Set the details of the output file using the `outputPath` field of the `AdvancedTransformerDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `AdvancedTransformerDetail` instance.

3. To create and run the Spark job, use the previously created instance of `DataNormalizationFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `AdvancedTransformerDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. Display the counters to view the reporting statistics for the job.

# Open Parser

Open Parser parses your input data from many cultures of the world using a simple but powerful parsing grammar. Using this grammar, you can define a sequence of expressions that represent domain patterns for parsing your input data. Open Parser also collects statistical data and scores the parsing matches to help you determine the effectiveness of your parsing grammars.

Use Open Name Parser to:

- Parse input data using domain-specific and culture-specific parsing grammars that you define in Domain Editor.
- Parse input data using domain-independent parsing grammars that you define in Open Parser using the same simple but powerful parsing grammar available in Domain Editor.
- Parse input data using domain-independent parsing grammars at runtime that you define in Dataflow Options.
- Preview parsing grammars to test how sample input data parses before running the job using the target input data file.
- Trace parsing grammar results to view how tokens matched or did not match the expressions you defined and to better understand the matching process.

## Open Parser Job

Open Parser job parses the input data strings according to the defined parsing grammar.

## API Entities

### *OpenParserDetail*

*Purpose*

To specify details of an Open Parser job.

### *OpenParserConfiguration*

*Purpose*

To break a string of input text into its component parts.

## Input Parameters

| Parameter | Description |
|---|---|
| Open Parser Configuration | To parse an input string into its constituting components. |
| Reference Data Path | To specify the Reference Data path details. |

| Parameter | Description |
|---|---|
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |

### Output Columns

The job breaks the input data strings into its constituents on the basis of the fields defined in the parsing grammar and displays those in the output columns. It also displays `Yes` or `No` to indicate if the record was parsed or not.

### Using an Open Parser MapReduce Job

1. Create an instance of `DataNormalizationFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Open Parser job by creating an instance of `OpenParserDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.
   a) Configure the parsing rules by creating an instance of `OpenParserConfiguration`. In this instance, set the grammar file path.
   b) Set the details of the Reference Data Path and location type by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.
   c) Create an instance of `OpenParserDetail`, by passing an instance of type `JobConfig`, and the `OpenParserConfiguration` and `ReferenceDataPath` instances created earlier as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.
   d) In the instance of the `OpenParserDetail` created above, set the details of the input file using the `inputPath` field of the `OpenParserDetail` instance.

- For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

e) Set the details of the output file using the `outputPath` field of the `OpenParserDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `OpenParserDetail` instance.

3. To create a MapReduce job, use the previously created instance of `DataNormalizationFactory` to invoke its method `createJob()`. In this, pass the above instance of `OpenParserDetail` as an argument.
   The `createJob()` method returns a `List` of instances of `ControlledJob`.
4. Run the created job using an instance of `JobControl`.

### Using an Open Parser Spark Job

1. Create an instance of `DataNormalizationFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Open Parser job by creating an instance of `OpenParserDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.
   a) Configure the parsing rules by creating an instance of `OpenParserConfiguration`. In this instance, set the grammar file path.
   b) Set the details of the Reference Data Path and location type by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.
   a) Create an instance of `OpenParserDetail`, by passing an instance of type `JobConfig`, and the `OpenParserConfiguration` and `ReferenceDataPath` instances created earlier as the arguments to its constructor.
      The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.
   a) In the instance of the `OpenParserDetail` created above, set the details of the input file using the `inputPath` field of the `OpenParserDetail` instance.
      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.

- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

b) Set the details of the output file using the `outputPath` field of the `OpenParserDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

c) Set the name of the job using the `jobName` field of the `OpenParserDetail` instance.

3. To create and run a Spark job, use the previously created instance of `DataNormalizationFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `OpenParserDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

## Table Lookup

The Table Lookup stage standardizes terms against a previously validated form of that term and applies the standard version. This evaluation is done by searching a table for the term to standardize.

### Table Lookup Job

The Table Lookup job standardizes terms against a previously validated form of that term and applies the standard version.

#### API Entities

##### *AbstractTableLookupRule*

*Purpose*

To specify the rule to be used for Table Lookup.

##### *Categorize*

*Purpose*

To specify the Categorize rule for a Table Lookup job.

### *Identify*

*Purpose*
To specify the Identify rule for a Table Lookup job.

### *Standardize*

*Purpose*
To specify the Standardize rule for a Table Lookup job.

### *TableLookupDetail*

*Purpose*
To specify details of a Table Lookup job.

### *TableLookupConfiguration*

*Purpose*
To standardize terms against a previously validated form of that term, and to apply the standardized version to all records.

## Input Parameters

| Parameter | Description |
|---|---|
| Table Lookup Configuration | To standardize terms against a previously validated form of that term, and to apply the standardized version to all records.<br>The rules can be of the type `Standardize`, `Categorize` or `Identify`. |
| Reference Data Path | To specify the Reference Data path details. |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |

| Parameter | Description |
|---|---|
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |
| Compress Output | Flag to indicate if the output must be compressed.<br>Set this to `true` to compress the output. |

## Output Columns

In addition to the input columns, the following columns are added while generating the output of a Table Lookup job:

| Column | Description | Output Value |
|---|---|---|
| Destination | For `Standardize` and `Categorize` rule options, this output column is added if a new column name, not present in the input, is specified as the destination column.<br>The name of the column is as entered by you.<br>**Note:** For the destination column, you can select an existing source column or type a new column name. | The standardized value of the source columns, matched against the table data. |
| Standardization Term Identified | Indicates whether the standardized term has been identified or not. | The possible value is `Yes` and `No`. |

## Using a Table Lookup MapReduce Job

1. Create an instance of `DataNormalizationFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Table Lookup job by creating an instance of `TableLookupDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.
   a) Configure the table lookup rules by creating an instance of `TableLookupConfiguration`.
   Within this instance, add an instance of type `AbstractTableLookupRule`. This `AbstractTableLookupRule` instance must be defined using one of these classes: `Standardize`, `Categorize` or `Identify`, corresponding to the desired table lookup rule category.
   b) Set the details of the Reference Data path and location type by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.
   c) Create an instance of `TableLookupDetail`, by passing an instance of type `JobConfig`, and the `TableLookupConfiguration` and `ReferenceDataPath` instances created earlier as the arguments to its constructor.
   The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.
   d) Set the details of the input file using the `inputPath` field of the `TableLookupDetail` instance.
   - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
   - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
   - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

e) Set the details of the output file using the `outputPath` field of the `TableLookupDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `TableLookupDetail` instance.

g) Set the `compressOutput` flag of the `TableLookupDetail` instance to `true` to compress the output of the job.

3. To create a MapReduce job, use the previously created instance of `DataNormalizationFactory` to invoke its method `createJob()`. In this, pass the above instance of `TableLookupDetail` as an argument.
   The `createJob()` method returns a `List` of instances of `ControlledJob`.
4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters post a successful MapReduce job run, use the previously created instance of `DataNormalizationFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using a Table Lookup Spark Job

1. Create an instance of `DataNormalizationFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Table Lookup job by creating an instance of `TableLookupDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.

   a) Configure the table lookup rules by creating an instance of `TableLookupConfiguration`.

   Within this instance, add an instance of type `AbstractTableLookupRule`. This `AbstractTableLookupRule` instance must be defined using one of these classes: `Standardize`, `Categorize` or `Identify`, corresponding to the desired table lookup rule category.

   b) Set the details of the Reference Data path and location type by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.

   c) Create an instance of `TableLookupDetail`, by passing an instance of type `JobConfig`, and the `TableLookupConfiguration` and `ReferenceDataPath` instances created earlier as the arguments to its constructor.

   The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.

   d) Set the details of the input file using the `inputPath` field of the `TableLookupDetail` instance.

- For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

e) Set the details of the output file using the `outputPath` field of the `TableLookupDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `TableLookupDetail` instance.

g) Set the `compressOutput` flag of the `TableLookupDetail` instance to `true` to compress the output of the job.

3. To create and run the Spark job, use the previously created instance of `DataNormalizationFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `TableLookupDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. Display the counters to view the reporting statistics for the job.

# Global Addressing Module Jobs

## Global Address Validation

Global Address Validation provides enhanced address standardization and validation. Global Address Validation is part of the Global Addressing Module.

## API Entities

### *AddressValidationDetail<T extends ProcessType>*

*Purpose*
To specify the details of a Global Address Validation job.

### *AddressValidationEngineConfiguration*

*Purpose*
To set the `ProductDatabaseInfoList` for the Global Address Validation job.

These are one-time settings.

### *AddressValidationFactory*

*Purpose*
A singleton factory class to create instances of Global Address Validation jobs.

This instance is used to generate the reporting counters.

### *ProductDatabaseInfo*

*Purpose*
To set the database path, country codes, and process type to create and run the Global Address Validation job.

### *AddressValidationInputOption*

*Purpose*
To configure settings for the input to create and run the Global Address Validation job. This is a rule setting, and has various options. These settings vary for every job.

## Input Parameters

| Parameter | Description |
|---|---|
| Address Validation Engine Configuration | To set the product database info list. |

| Parameter | Description |
| --- | --- |
| Address Validation Input Option | To configure the input settings:<br>1. Output Casing<br>2. Match Mode<br>3. Default Country<br>4. Maximum Results<br>5. Return Input Address<br>6. Return Parsed Address<br>7. Return Precision Code<br>8. Return Match Score<br>9. Must Match Address Number<br>10. Must Match Street<br>11. Must Match City<br>12. Must Match Locality<br>13. Must Match State<br>14. Must Match State Province<br>15. Must Match Post Code<br>16. Keep Multi Match<br>17. Prefer Postal Over City<br>18. City Fallback<br>19. Postal Fallback<br>20. Validation Level |
| Product Database Info | To set:<br>1. Country Code<br>2. Database Path<br>3. Process Type |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |

| Parameter | Description |
| --- | --- |
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |
| Compress Output | Flag to indicate if the output must be compressed.<br>Set this to `true` to compress the output. |

## Output Columns

1. AddressBlock1-2
2. ApartmentLabel
3. ApartmentNumber
4. Building
5. City
6. City.Matched
7. CitySubdivision

8. CitySubdivision.Matched
9. Confidence
10. Country
11. Country specific fields
12. FirmName
13. HouseNumber
14. Housenumber.Matched
15. LeadingDirectional
16. MatchOnAllStreetFields
17. MatchOnStreetDirectional
18. MultimatchCount
19. PostalCode
20. PostalCode.AddOn
21. Postalcode.Matched
22. Principality
23. ProcessedBy
24. StateProvince
25. StateProvince.Matched
26. StateProvinceSubdivision
27. StateProvinceSubdivision.Matched
28. StreetName
29. StreetName.Matched
30. StreetType
31. StreetType.Matched
32. TrailingDirectional
33. Vendor Code

**Note:** For field descriptions, see the topic *Global Address Validation* in the *Addressing Guide* of Spectrum™ Technology Platform.

### Using a Global Address Validation MapReduce Job

1. Create an instance of `AddressValidationFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Global Address Validation job by creating an instance of `AddressValidationDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32. For this, the steps are:
   a) Create an instance of `productDatabaseInfo`, and set these details:
      1. ReferenceDataPath: Use **Enum ReferenceDataPathLocation** on page 357
      2. CountryCode: Use **Enum CountryCodes** on page 360
      3. ProcessType: Use **Enum AddressValidationProcessType** on page 367

b) Create an array list class `ProductDatabaseInfoList` and use the add() method to insert the `ProductDatabaseInfo`.
c) Create an instance of `AddressValidationEngineConfiguration`, and in this instance, set the `ProductDatabaseInfoList`.
d) Create an instance of `AddressValidationInputOption`, and set these details to this new instance:

> **Note:** Use these enums: **Enum AddressValidationInputOption.MatchMode** on page 367, **Enum CountryCodes** on page 360, and **Enum Casing** on page 361.

- Casing
- MatchMode
- DefaultCountry
- MaximumResults
- ReturnInputAddress
- ReturnParsedAddress
- ReturnPrecisionCode
- ReturnMatchScore
- MustMatchAddressNumber
- MustMatchStreet
- MustMatchCity
- MustMatchLocality
- MustMatchState
- MustMatchStateProvince
- MustMatchPostCode
- KeepMultiMatch
- PreferPostalOverCity
- CityFallback
- PostalFallback
- ValidationLevel

e) Create an instance of `AddressValidationDetail`, by passing the job configuration, `addressValidationEngineConfiguration`, and `inputOption` instance created earlier as the arguments to its constructor. To this instance, set these details:

> **Note:** The `Config` parameter must be an instance of type **MRJobConfig** on page 30 (for an MR job) and **SparkJobConfig** on page 31 (for a Spark job).

1. Set the details of the input file using the `inputPath` field.

> **Note:**
>
> - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.

- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

2. Set the details of the output file using the `outputPath` field.

   **Note:**

   - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
   - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
   - For a parquet output file, create an instance of `ParquetFilePath` with the path of the parquet output file as the argument.

3. Set the name of the job using the `jobName` field.
4. Set the `compressOutput` flag to false to prevent compressing the output of the job.

3. To create a MapReduce job, use the previously created instance of `AddressValidationFactory` to invoke its method `createJob()`. In this, pass the above instance of `AddressValidationDetail` as an argument.
   The `createJob()` method returns a `List` of instances of `ControlledJob`.
4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters post a successful MapReduce job run, use the previously created instance of `AddressValidationFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using a Global Address Validation Spark Job

1. Create an instance of `AddressValidationFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Global Address Validation job by creating an instance of `AddressValidationDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32. For this, the steps are:
   a) Create an instance of `productDatabaseInfo`, and set these details:
      1. ReferenceDataPath: Use **Enum ReferenceDataPathLocation** on page 357
      2. CountryCode: Use **Enum CountryCodes** on page 360
      3. ProcessType: Use **Enum AddressValidationProcessType** on page 367

   b) Create an array list class `ProductDatabaseInfoList` and use the add() method to insert the `ProductDatabaseInfo`.
   c) Create an instance of `AddressValidationEngineConfiguration`, and in this instance, set the `ProductDatabaseInfoList`.

d) Create an instance of `AddressValidationInputOption`, and set these details to this new instance:

> **Note:** Use these enums: **Enum AddressValidationInputOption.MatchMode** on page 367, **Enum CountryCodes** on page 360, and **Enum Casing** on page 361.

- Casing
- MatchMode
- DefaultCountry
- MaximumResults
- ReturnInputAddress
- ReturnParsedAddress
- ReturnPrecisionCode
- ReturnMatchScore
- MustMatchAddressNumber
- MustMatchStreet
- MustMatchCity
- MustMatchLocality
- MustMatchState
- MustMatchStateProvince
- MustMatchPostCode
- KeepMultiMatch
- PreferPostalOverCity
- CityFallback
- PostalFallback
- ValidationLevel

e) Create an instance of `AddressValidationDetail`, by passing the job configuration, `addressValidationEngineConfiguration`, and `inputOption` instance created earlier as the arguments to its constructor. To this instance, set these details:

> **Note:** The `Config` parameter must be an instance of type **MRJobConfig** on page 30 (for an MR job) and **SparkJobConfig** on page 31 (for a Spark job).

1. Set the details of the input file using the `inputPath` field.

   **Note:**

   - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
   - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
   - For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

2. Set the details of the output file using the `outputPath` field.

   **Note:**

   - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
   - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
   - For a parquet output file, create an instance of `ParquetFilePath` with the path of the parquet output file as the argument.

3. Set the name of the job using the `jobName` field.
4. Set the `compressOutput` flag to false to prevent compressing the output of the job.

3. To create a Spark job, use the previously created instance of `AddressValidationFactory` to invoke its method `createJob()`. In this, pass the above instance of `AddressValidationDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.
4. Display the counters to view the reporting statistics for the job.

# Universal Addressing Module Jobs

## Common Module APIs

### UniversalAddressingDetail<T extends ProcessType>

*Purpose*
To specify the details of a **Universal Addressing Module** job.

### UniversalAddressingFactory

*Purpose*
A singleton factory class to create instances of **Universal Addressing Module** jobs.

# Validate Address

Validate Address standardizes and validates addresses using postal authority address data. It can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state or province names, and more.

> **Note:** Currently, Validate Address supports only US addresses.

Validate Address also returns result indicators about validation attempts, such as whether it validated the address, the level of confidence in the returned address, and the reason for failure if the address could not be validated.

During address matching and standardization, Validate Address separates address lines into components and compares those to the contents of the **Universal Addressing Module** databases. If a match is found, the input address is *standardized* to the database information. If no database match is found, Validate Address optionally *formats* the input addresses. The formatting process attempts to structure the address lines according to the conventions of the appropriate postal authority.

> **Note:** Validate Address supports CASS Certified™ processing which enables you to qualify for USPS® postal discounts.

## API Entities

### *UAMAddressingDetail<T extends ProcessType>*

*Purpose*
To specify the details of a Validate Address job.

### *UniversalAddressEngineConfiguration*

*Purpose*
To set various configurations like the *reference data path* and *COBOL runtime path* required to create and run the Validate Address job.

These are one-time settings.

### *UAMAddressingFactory*

*Purpose*
A singleton factory class to create instances of Validate Address jobs.

This instance is used to generate the reporting counters, and the CASS reports.

### *UniversalAddressGeneralConfiguration*

*Purpose*
To set JVM configurations required to create and run the Validate Address job.

### *UniversalAddressValidateInputConfiguration*

*Purpose*
To configure settings for the input to create and run the Validate Address job. This is a rule setting, and has various options. These settings vary for every job.

## Input Parameters

| Parameter | Description |
|---|---|
| Universal Address Engine Configuration | To set various job run configurations:<br>1. DPV Database Path<br>2. Suite Link DB Path<br>3. EWS Database Path<br>4. RDI Database Path<br>5. Lacs Database Path<br>6. Reference Data Path<br>7. COBOL Runtime Path<br>8. Modules directory |

| Parameter | Description |
| --- | --- |
| Universal Address Validate Input Configuration | |

| Parameter | Description |
| --- | --- |
| | To configure the input settings:<br>1. Output Standard Address<br>2. Output Address Elements<br>3. Output Postal Data<br>4. Output Parsed Input<br>5. Output Address Blocks<br>6. Output Formatted On Fail<br>7. Output Casing<br>8. Output Postal Code Separator<br>9. Output Multinational Characters<br>10. Perform DPV<br>11. Perform RDI<br>12. Perform ESM<br>13. Perform ASM<br>14. Perform EWS<br>15. Perform LACS Link<br>16. Perform LOT<br>17. Fail On CMRA Match<br>18. Extract Firm<br>19. Extract Urb<br>20. Output Report 3553<br>21. Output Report SERP<br>22. Output Report Summary<br>23. Output CASS Detail<br>24. Output Field Level Return Codes<br>25. Keep Multimatch<br>26. Maximum Results<br>27. Standard Address Format<br>28. Standard Address PMB Line<br>29. City Name Format<br>30. Vanity City Format Long<br>31. Output Country Format<br>32. Home Country<br>33. Street Matching Strictness<br>34. Firm Matching Strictness<br>35. Directional Matching Strictness<br>36. Dual Address Logic<br>37. DPV Successful Status Condition<br>38. Report List File Name<br>39. Report List Processor Name<br>40. Report List Number<br>41. Report Mailer Address<br>42. Report Mailer Name<br>43. Report Mailer City Line<br>44. Address Line Search On Fail<br>45. Output Street Alias |

| Parameter | Description |
| --- | --- |
| | 46. Output VeriMove Block<br>47. DPV Determine No Stat<br>48. DPV Determine Vacancy<br>49. Output Abbreviated Alias<br>50. Output Preferred Alias<br>51. Output Preferred City<br>52. Perform Suite Link<br>53. Suppress Zplus Phantom Carrier R777 |
| Universal Address General Configuration | To set JVM configurations:<br>1. DPV File Type<br>2. DPV Memory Model<br>3. Lacs Link Memory Model<br>4. Suite Link Memory Model |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |

| Parameter | Description |
|---|---|
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |
| Compress Output | Flag to indicate if the output must be compressed.<br>Set this to `true` to compress the output. |

| Parameter | Description |
|---|---|
| CASS Reports | The configurations to generate the CASS report. Invoke any of the overloaded methods `generateCASSReport()` using the `UAMAddressingFactory` instance.<br><br>The CASS reports are generated in PDF format.<br><br>The parameters are:<br><br>**Counters** — A Map of the counters to be included in the CASS report.<br><br>**Job Name** — The name of the job. This is included in the filename of the CASS report.<br><br>**Path** — The directory where the created CASS report is placed. This is an optional input value for the CASS reports.<br><br>The `path` must be on the cluster or client location depending on whether the SDK job is running in a cluster environment or on your client machine, respectively.<br><br>**Note:** If the `path` is not specified, the new CASS report is placed in the current working directory.<br><br>**Report Type** — The type of CASS report to be generated. You can specify one or more values from **Enum UAMCASSReportType** on page 367. |

## Output Columns

1. AdditionalInputData
2. AddressLine1
3. AddressLine2
4. AddressLine3
5. AddressLine4
6. AddressLine5
7. City
8. Country
9. FirmName
10. PostalCode
11. PostalCode.AddOn
12. PostalCode.Base
13. StateProvince
14. USUrbanName
15. AdditionalInputData
16. ApartmentLabel
17. ApartmentLabel2
18. ApartmentNumber
19. ApartmentNumber2

20. HouseNumber
21. LeadingDirectional
22. POBox
23. PrivateMailbox
24. PrivateMailbox.Type
25. RRHC
26. StateProvince
27. StreetName
28. StreetSuffix
29. TrailingDirectional
30. USUrbanName
31. ApartmentLabel.Input
32. ApartmentNumber.Input
33. City.Input
34. Country.Input
35. FirmName.Input
36. HouseNumber.Input
37. LeadingDirectional.Input
38. POBox.Input
39. PostalCode.Input
40. PrivateMailbox.Input
41. PrivateMailbox.Type.Input
42. RRHC.Input
43. StateProvince.Input
44. StreetName.Input
45. StreetSuffix.Input
46. TrailingDirectional.Input
47. USUrbanName.Input
48. PostalBarCode
49. USAltAddr
50. USBCCheckDigit
51. USCarrierRouteCode
52. USCongressionalDistrict
53. USCountyName
54. USFinanceNumber
55. USFIPSCountyNumber
56. USLACS
57. USLastLineNumber
58. AddressFormat
59. Confidence
60. CouldNotValidate

61. CountryLevel
62. MatchScore
63. MultimatchCount
64. MultipleMatches
65. ProcessedBy
66. RecordType
67. RecordType.Default
68. Status
69. Status.Code
70. Status.Description
71. AddressRecord.Result
72. ApartmentLabel.Result
73. ApartmentNumber.Result
74. City.Result
75. Country.Result
76. FirmName.Result
77. HouseNumber.Result
78. LeadingDirectional.Result
79. POBox.Result
80. PostalCode.Result
81. PostalCodeCity.Result
82. PostalCode.Source
83. PostalCode.Type
84. RRHC.Result
85. RRHC.Type
86. StateProvince.Result
87. Street.Result
88. StreetName.AbbreviatedAlias.Result
89. StreetName.Alias.Type
90. StreetName.PreferredAlias.Result
91. StreetName.Result
92. StreetSuffix.Result
93. TrailingDirectional.Result
94. USUrbanName.Result
95. USLOTCode
96. USLOTHex
97. USLOTSequence
98. USLACS.ReturnCode
99. RDI
100. DPV
101. CMRA

102. DPVFootnote
103. DPVVacant
104. DPVNoStat
105. SuiteLinkReturnCode
106. SuiteLinkMatchCode
107. SuiteLinkFidelity
108. VeriMoveDataBlock

**Note:** For the field descriptions, see the topic *Validate Address* in the *Addressing Guide* of Spectrum™ Technology Platform.

## Using a Validate Address MapReduce Job

**Attention:** Before creating and running the first Validate Address job, ensure the Acushare service is running. For steps, see **Running Acushare Service** on page 13.

1. Create an instance of `UAMAddressingFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Validate Address job by creating an instance of `UAMAddressingDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32. For this, the steps are:
   a) To configure the input settings for the job, create an instance of `UniversalAddressValidateInputConfiguration`.

   Set the values of the various required fields of this instance, using the enums **Enum PreferredCity** on page 366, **Enum CasingType** on page 364, **Enum CityNameFormat** on page 364, **Enum OutputCountryFormat** on page 364, **Enum StandardAddressFormat** on page 365, **Enum StandardAddressPMBLine** on page 365, **Enum StreetMatchingStrictness** on page 365, **Enum FirmMatchingStrictness** on page 365, **Enum DirectionalMatchingStrictness** on page 365, **Enum DualAddressLogic** on page 364, and **Enum DPVSuccessStatusCondition** on page 367 where applicable.

   **Important:** To run Validate Address in the CASS Certified™ mode, set the fields `outputReport3553`, `outputCASSDetail`, and `outputReportSummary` of this instance to `true`. The CASS reports contain valid content only when the job is run in the CASS Certified™ mode. Else, blank report PDFs are generated.

   b) Set the details of the *Reference Data path* by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.
   c) To configure the various job run settings, create an instance of `UAMUSAddressingEngineConfiguration` by passing the `ReferenceDataPath` instance created above, and the *COBOL Runtime path* and *modules directory path* as `String` values, as arguments to its constructor.

   Once the `UAMUSAddressingEngineConfiguration` instance is created, set the values for its various required fields.

   d) To configure JVM settings, create an instance of `UniversalAddressGeneralConfiguration`.

Use the enums **Enum DPVFileType** on page 366, **Enum DPVMemoryModel** on page 366, **Enum LacsLinkMemoryModel** on page 366, and **Enum SuiteLinkMemoryModel** on page 366.

e) Create an instance of `UAMAddressingDetail`, by passing an instance of type `JobConfig`, and the instances of `UAMUSAddressingEngineConfiguration`, `UniversalAddressGeneralConfiguration`, and `UniversalAddressValidateInputConfiguration` created above as the arguments to its constructor.

The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.

1. Set the details of the input file using the `inputPath` field of the `UAMAddressingDetail` instance.

   **Note:**

   - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
   - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
   - For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

2. Set the details of the output file using the `outputPath` field of the `UAMAddressingDetail` instance.

   **Note:**

   - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
   - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
   - For a parquet output file, create an instance of `ParquetFilePath` with the path of the parquet output file as the argument.

3. Set the name of the job using the `jobName` field of the `UAMAddressingDetail` instance.
4. Set the `compressOutput` flag of the `UAMAddressingDetail` instance to `true` to compress the output of the job.

3. To create a MapReduce job, use the previously created instance of `UAMAddressingFactory` to invoke its method `createJob()`. In this, pass the above instance of `UAMAddressingDetail` as an argument.
   The `createJob()` method returns a `List` of instances of `ControlledJob`.
4. Run the created job using an instance of `JobControl`.

5. To display the reporting counters post a successful job run, use the previously created instance of `UAMAddressingFactory` to invoke its method `getCounters()`, passing the created job as an argument.
   A `Map` of counters is received.
6. To generate the CASS reports after a successful job run, use the previously created instance of `UAMAddressingFactory` to invoke the method `generateCASSReport()`. You can invoke any of the overloaded versions of the method `generateCASSReport()`.

   Depending on which `generateCASSReport()` method signature is used, pass as arguments the `Map` of reporting counters derived in the previous step, the `jobName`, the `path` where the generated CASS report must be stored, and the required `reportType` to be created.

   The `path` must be on the cluster or client location depending on whether the SDK job is running in a cluster environment or on your client machine, respectively.

   > **Note:** If the `path` is not specified, the new CASS report is placed in the current working directory.

   The `reportType` parameter must have values from the **Enum UAMCASSReportType** on page 367. You can specify one or more report types in this parameter.

### Using a Validate Address Spark Job

**Attention:** Before creating and running the first Validate Address job, ensure the Acushare service is running. For steps, see **Running Acushare Service** on page 13.

1. Create an instance of `UAMAddressingFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Validate Address job by creating an instance of `UAMAddressingDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32. For this, the steps are:
   a) To configure the input settings for the job, create an instance of `UniversalAddressValidateInputConfiguration`.

      Set the values of the various required fields of this instance, using the enums **Enum PreferredCity** on page 366, **Enum CasingType** on page 364, **Enum CityNameFormat** on page 364, **Enum OutputCountryFormat** on page 364, **Enum StandardAddressFormat** on page 365, **Enum StandardAddressPMBLine** on page 365, **Enum StreetMatchingStrictness** on page 365, **Enum FirmMatchingStrictness** on page 365, **Enum DirectionalMatchingStrictness** on page 365, **Enum DualAddressLogic** on page 364, and **Enum DPVSuccessStatusCondition** on page 367 where applicable.

      **Important:** To run Validate Address in the CASS Certified™ mode, set the fields `outputReport3553`, `outputCASSDetail`, and `outputReportSummary` of this instance to `true`. The CASS reports contain valid content only when the job is run in the CASS Certified™ mode. Else, blank report PDFs are generated.

   b) Set the details of the *Reference Data path* by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.

c) To configure the various job run settings, create an instance of `UAMUSAddressingEngineConfiguration` by passing the `ReferenceDataPath` instance created above, and the *COBOL Runtime path* and *modules directory path* as `String` values, as arguments to its constructor.

   Once the `UAMUSAddressingEngineConfiguration` instance is created, set the values for its various required fields.

d) To configure JVM settings, create an instance of `UniversalAddressGeneralConfiguration`.

   Use the enums **Enum DPVFileType** on page 366, **Enum DPVMemoryModel** on page 366, **Enum LacsLinkMemoryModel** on page 366, and **Enum SuiteLinkMemoryModel** on page 366.

e) Create an instance of `UAMAddressingDetail`, by passing an instance of type `JobConfig`, and the instances of `UAMUSAddressingEngineConfiguration`, `UniversalAddressGeneralConfiguration`, and `UniversalAddressValidateInputConfiguration` created above as the arguments to its constructor.

   The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.

   1. Set the details of the input file using the `inputPath` field of the `UAMAddressingDetail` instance.

      **Note:**

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
      - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
      - For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

   2. Set the details of the output file using the `outputPath` field of the `UAMAddressingDetail` instance.

      **Note:**

      - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
      - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
      - For a parquet output file, create an instance of `ParquetFilePath` with the path of the parquet output file as the argument.

   3. Set the name of the job using the `jobName` field of the `UAMAddressingDetail` instance.

4. Set the `compressOutput` flag of the `UAMAddressingDetail` instance to `true` to compress the output of the job.

3. To create and run the Spark job, use the previously created instance of `UAMAddressingFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `UAMAddressingDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. To display the reporting counters post a successful job run, use the previously created instance of `UAMAddressingFactory` to invoke its method `getCounters()`, passing the created job as an argument.
   A `Map` of counters is received.

5. To generate the CASS reports after a successful job run, use the previously created instance of `UAMAddressingFactory` to invoke the method `generateCASSReport()`. You can invoke any of the overloaded versions of the method `generateCASSReport()`.

   Depending on which `generateCASSReport()` method signature is used, pass as arguments the `Map` of reporting counters derived in the previous step, the `jobName`, the `path` where the generated CASS report must be stored, and the required `reportType` to be created.

   The `path` must be on the cluster or client location depending on whether the SDK job is running in a cluster environment or on your client machine, respectively.

   > **Note:** If the `path` is not specified, the new CASS report is placed in the current working directory.

   The `reportType` parameter must have values from the **Enum UAMCASSReportType** on page 367. You can specify one or more report types in this parameter.

## Validate Address Global

Validate Address Global provides enhanced address standardization and validation for addresses outside the U.S. and Canada. Validate Address Global can also validate addresses in the U.S. and Canada but its strength is validation of addresses in other countries. If you process a significant number of addresses outside the U.S. and Canada, you should consider using Validate Address Global.

Validate Address Global is part of the Universal Addressing Module.

Validate Address Global performs several steps to achieve a quality address, including parsing, validation, and formatting.

### *Address Parsing, Formatting, and Standardization*

Restructuring incorrectly fielded address data is a complex and difficult task especially when done for international addresses. People introduce many ambiguities as they enter address data into computer systems. Among the problems are misplaced elements (such as company or personal names in street address fields) or varying abbreviations that are not only language, but also country specific. Validate Address Global identifies address elements in address lines and assigns them to the proper fields. This is an important precursor to the actual validation. Without restructuring, "no match" situations might result.

Properly identified address elements are also important when addresses have to be truncated or shortened to fit specific field length requirements. With the proper information in the right fields, specific truncation rules can be applied.

- Parses and analyzes address lines and identifies individual address elements
- Processes over 30 different character sets
- Formats addresses according to the postal rules of the country of destination
- Standardizes address elements (such as changing AVENUE to AVE)

### *Global Address Validation*

Address validation is the correction process where properly parsed address data is compared against reference databases supplied by postal organizations or other data providers. Validate Address Global validates individual address elements to check for correctness using sophisticated fuzzy matching technology and produces standardized and formatted output based on postal standards and user preferences. FastCompletion validation type can be used in quick address entry applications. It allows input of truncated data in several address fields and generates suggestions based on this input.

In some cases, it is not possible to fully validate an address. Here Validate Address Global has a unique deliverability assessment feature that classifies addresses according to their probable deliverability.

## API Entities

#### ***GlobalAddressingDetail<T extends ProcessType>***

*Purpose*
To specify the details of a Validate Address Global job.

#### ***GlobalAddressingEngineConfiguration***

*Purpose*
To set database configurations required to create and run the Validate Address Global job.

#### ***GlobalAddressingFactory***

*Purpose*
A singleton factory class to create instances of Validate Address Global jobs.

### *GlobalAddressingGeneralConfiguration*

*Purpose*
To set JVM configurations required to create and run the Validate Address Global job.

### *GlobalAddressingInputConfiguration*

*Purpose*
To configure settings for the input to create and run the Validate Address Global job.

## Input Parameters

| Parameter | Description |
|---|---|
| Validate Address Global Engine Configuration | To set database configurations:<br>1. Database Type<br>2. Preloading Type<br>3. Reference Data Path<br>4. If all countries are supported. If not, list of supported Countries |
| Validate Address Global Input Configuration | To configure these settings for the input:<br>1. State Province Type in result<br>2. Matching Scope in process<br>3. Force Country ISO3 in input<br>4. Default Country ISO3 in input<br>5. Format Delimiter in input<br>6. Format Delimiter in result<br>7. Include inputs in result<br>8. Country Type in result<br>9. Optimization Level of process<br>10. Preferred Language of result<br>11. Mode of process<br>12. Preferred Script in result<br>13. Maximum Results<br>14. Casing of result |
| Validate Address Global General Configuration | To set JVM configurations:<br>1. Cache Size<br>2. Maximum Thread Count<br>3. Maximum Address Object Count<br>4. Ranges to expand<br>5. Flexible Range Expansion<br>6. Enable Transaction Logging<br>7. Maximum Memory Usage in MB |

| Parameter | Description |
| --- | --- |
| Unlock Code | To unlock the data in the database. |
| Reference Data Path | To specify the Reference Data path details.<br>For the UAM jobs, reference data can be placed on the local data nodes in the cluster or on HDFS.<br>**Note:** In case of local data nodes the reference data needs to be placed as un-archived folders while for HDFS these need to be archived files in `.zip` format. |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |

| Parameter | Description |
|---|---|
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |

## Output Columns

### *Address Data*

1. AddressBlock1-9
2. AddressLine1-6
3. AdministrativeDistrict
4. ApartmentLabel
5. ApartmentNumber
6. BlockName
7. BuildingName
8. City
9. City.AddInfo

10. City.SortingCode
11. Contact
12. Country
13. County
14. FirmName
15. Floor
16. HouseNumber
17. LastLine
18. LeadingDirectional
19. Locality
20. POBox
21. PostalCode
22. PostalCode.AddOn
23. PostalCode.Base
24. Room
25. SecondaryStreet
26. StateProvince
27. StreetName
28. StreetSuffix
29. SubBuilding
30. Suburb
31. Territory
32. TrailingDirectional

### *Original Input Data*

1. AddressLine1.Input
2. AddressLine2.Input
3. AddressLine3.Input
4. AddressLine4.Input
5. AddressLine5.Input
6. AddressLine6.Input
7. City.Input
8. StateProvince.Input
9. PostalCode.Input
10. Contact.Input
11. Country.Input
12. FirmName.Input
13. Street.Input
14. Number.Input
15. Building.Input
16. SubBuilding.Input

17. DeliveryService.Input

**Attention:** The input fields `AddressLine2.Input`, `AddressLine3.Input`, `AddressLine4.Input`, `AddressLine5.Input`, and `AddressLine6.Input` are included in the output only if the `resultIncludeInputs` field of the class `GlobalAddressingInputConfiguration` is set to `true`. Else, only those `AddressLineX.input` fields are included in output which are part of the input.

### *Result Codes*

1. AddressType
2. Confidence
3. CountOverflow
4. ElementInputStatus
5. ElementRelevance
6. ElementResultStatus
7. MailabilityScore
8. ModeUsed
9. MultimatchCount
10. ProcessStatus
11. Status
12. Status.Code
13. Status.Description

**Note:** For the field descriptions, see the *Validate Address Global* topic of the *Addressing Guide* of Spectrum™ Technology Platform.

## Using a Validate Address Global MapReduce Job

1. Create an instance of `GlobalAddressingFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Validate Address Global job by creating an instance of `GlobalAddressingDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32. For this, the steps are:
   a) Configure the JVM initialization settings by creating an instance of `GlobalAddressingGeneralConfiguration`.

      Use the enums **Enum CacheSize** on page 363, **Enum RangesToExpand** on page 364, and **Enum FlexibleRangeExpansion** on page 364.

   b) Set the details of the Reference Data path by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.
   c) Configure the necessary database settings by creating an instance of `GlobalAddressingEngineConfiguration` by passing the above `ReferenceDataPath` instance as an argument.

1. Set the *preloading type* in this instance using the enum **Enum PreloadingType** on page 360.
2. Set the *database type* using the **Enum DatabaseType** on page 360.
3. Set the supported countries using the **Enum CountryCodes** on page 360.
4. If all countries are supported, set the `isAllCountries` attribute to true. Else, specify the comma-separated list of **Enum CountryCodes** on page 360 values in the `supportedCountries` String value.

d) Configure the input settings by creating an instance of `GlobalAddressingInputConfiguration`.

To set the values of the various fields of this instance, use the enums **Enum CountryCodes** on page 360, **Enum StateProvinceType** on page 360, **Enum CountryType** on page 360, **Enum PreferredScript** on page 361, **Enum PreferredLanguage** on page 361, **Enum Casing** on page 361, **Enum OptimizationLevel** on page 362, **Enum Mode** on page 362, and **Enum MatchingScope** on page 362 as applicable.

e) Set the unlock key for the data as a `String` value in a `List`.

f) Create an instance of `GlobalAddressingDetail`, by passing an instance of type `Config`, the `List` of unlock code values, the `GlobalAddressingEngineConfiguration` instance, and the `GlobalAddressingInputConfiguration` instance created earlier as the arguments to its constructor.

The `Config` parameter must be an instance of type **MRJobConfig** on page 30.

The value of `GROUPBY_REGION` in this parameter is set to `true` by default. The jobs process the addresses of those regions for which you have added the reference data. For example, the input addresses of Germany are processed if reference data of Germany is placed on HDFS.

1. Set the JVM initialization configurations by setting the `generalConfiguration` field of the `GlobalAddressingDetail` instance to the `GlobalAddressingGeneralConfiguration` instance created above.
2. Set the details of the input file using the `inputPath` field of the `GlobalAddressingDetail` instance.

   **Note:**

   - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
   - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
   - For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

3. Set the details of the output file using the `outputPath` field of the `GlobalAddressingDetail` instance.

**Note:**

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a parquet output file, create an instance of `ParquetFilePath` with the path of the parquet output file as the argument.

4. Set the name of the job using the `jobName` field of the `GlobalAddressingDetail` instance.

3. To create a MapReduce job, use the previously created instance of `GlobalAddressingFactory` to invoke its method `createJob()`. In this, pass the above instance of `GlobalAddressingDetail` as an argument.
   The `createJob()` method returns a `List` of instances of `ControlledJob`.
4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters post a successful MapReduce job run, use the previously created instance of `GlobalAddressingFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using a Validate Address Global Spark Job

1. Create an instance of `GlobalAddressingFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Validate Address Global job by creating an instance of `GlobalAddressingDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32. For this, the steps are:
   a) Configure the JVM initialization settings by creating an instance of `GlobalAddressingGeneralConfiguration`.

      Use the enums **Enum CacheSize** on page 363, **Enum RangesToExpand** on page 364, and **Enum FlexibleRangeExpansion** on page 364.

   b) Set the details of the Reference Data path by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.
   c) Configure the necessary database settings by creating an instance of `GlobalAddressingEngineConfiguration` by passing the above `ReferenceDataPath` instance as an argument.

      1. Set the *preloading type* in this instance using the enum **Enum PreloadingType** on page 360.
      2. Set the *database type* using the **Enum DatabaseType** on page 360.
      3. Set the supported countries using the **Enum CountryCodes** on page 360.

4. If all countries are supported, set the `isAllCountries` attribute to true. Else, specify the comma-separated list of **Enum CountryCodes** on page 360 values in the `supportedCountries` String value.

d) Configure the input settings by creating an instance of `GlobalAddressingInputConfiguration`.

To set the values of the various fields of this instance, use the enums **Enum CountryCodes** on page 360, **Enum StateProvinceType** on page 360, **Enum CountryType** on page 360, **Enum PreferredScript** on page 361, **Enum PreferredLanguage** on page 361, **Enum Casing** on page 361, **Enum OptimizationLevel** on page 362, **Enum Mode** on page 362, and **Enum MatchingScope** on page 362 as applicable.

e) Set the unlock key for the data as a `String` value in a `List`.

f) Create an instance of `GlobalAddressingDetail`, by passing an instance of type `Config`, the `List` of unlock code values, the `GlobalAddressingEngineConfiguration` instance, and the `GlobalAddressingInputConfiguration` instance created earlier as the arguments to its constructor.

The `Config` parameter must be an instance of type **SparkJobConfig** on page 31.

The value of `GROUPBY_REGION` in this parameter is set to `true` by default. The jobs process the addresses of those regions for which you have added the reference data. For example, the input addresses of Germany are processed if reference data of Germany is placed on HDFS.

1. Set the JVM initialization configurations by setting the `generalConfiguration` field of the `GlobalAddressingDetail` instance to the `GlobalAddressingGeneralConfiguration` instance created above.
2. Set the details of the input file using the `inputPath` field of the `GlobalAddressingDetail` instance.

   **Note:**

   - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
   - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
   - For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

3. Set the details of the output file using the `outputPath` field of the `GlobalAddressingDetail` instance.

   **Note:**

   - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.

- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a parquet output file, create an instance of `ParquetFilePath` with the path of the parquet output file as the argument.

4. Set the name of the job using the `jobName` field of the `GlobalAddressingDetail` instance.

3. To create and run the Spark job, use the previously created instance of `GlobalAddressingFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `GlobalAddressingDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. Display the counters to view the reporting statistics for the job.

## Validate Address Loqate

Validate Address Loqate standardizes and validates addresses using postal authority address data. Validate Address Loqate can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state/province names.

Validate Address Loqate also returns result indicators about validation attempts, such as whether or not Validate Address Loqate validated the address, the level of confidence in the returned address, the reason for failure if the address could not be validated, and more.

During address matching and standardization, Validate Address Loqate separates address lines into components and compares them to the contents of the Universal Addressing Module databases. If a match is found, the input address is standardized to the database information. If no database match is found, ValidateAddress Loqate optionally formats the input addresses. The formatting process attempts to structure the address lines according to the conventions of the appropriate postal authority.Validate Address Loqate is part of the Universal Addressing Module.

### API Entities

***LoqateAddressingDetail<T extends ProcessType>***

*Purpose*
To specify the details of a Validate Address Loqate job.

***LoqateAddressingEngineConfiguration***

*Purpose*
To set database configurations required to create and run the Validate Address Loqate job.

### *LoqateAddressingFactory*

*Purpose*
A singleton factory class to create instances of Validate Address Loqate jobs.

### *LoqateAddressingGeneralConfiguration*

*Purpose*
To set JVM configurations required to create and run the Validate Address Loqate job.

### *LoqateAddressingValidateConfiguration*

*Purpose*
To configure settings for the input to create and run the Validate Address Loqate job.

## Input Parameters

| Parameter | Description |
|---|---|
| Validate Address Loqate Engine Configuration | To set configurations for performing the validations:<br>1. Verbose<br>2. Tool Info<br>3. Output Address Format<br>4. Log Input<br>5. Log Output<br>6. Log File Name<br>7. Match Score Absolute Threshold<br>8. Match Score Threshold Factor<br>9. Postal Code Max Results<br>10. Strict Reference Match |

| Parameter | Description |
| --- | --- |
| Validate Address Loqate Validate Configuration | To configure these settings for the input:<br>1. Include Standard Address<br>2. Include Matched Address Elements<br>3. Standardized Input Address Elements<br>4. Return Address Data Blocks<br>5. Output Casing<br>6. Include Result Codes for Individual Fields<br>7. Return Multiple Addresses<br>8. Failed On Multi Match Found<br>9. Multiple Address Count<br>10. Country Format<br>11. Default Country<br>12. Script Alphabet<br>13. Return Geocoded Address Fields<br>14. Acceptance Level<br>15. Minimum Match Score<br>16. Format Data Using AMAS Conventions<br>17. Is Duplicate Handling<br>18. Single Field Duplicate Handling<br>19. Multi Field Duplicate Handling<br>20. Non Standard Field Duplicate Handling<br>21. Output Field Duplicate Handling |
| Validate Address Loqate General Configuration | To set JVM configurations:<br>1. Maximum Idle Objects<br>2. Minimum Idle Objects<br>3. Maximum Active Objects<br>4. Maximum Wait Time<br>5. Action When Exhausted<br>6. Test on Borrow<br>7. Test on Return<br>8. Test While Idle<br>9. Time Between Eviction Runs in Milliseconds<br>10. Number of Tests Per Eviction Run<br>11. Min Evictable Idle Time in Milliseconds |
| Reference Data Path | To specify the Reference Data path details.<br>For the UAM jobs, reference data can be placed on the local data nodes in the cluster or on HDFS.<br>**Note:** In case of local data nodes the reference data needs to be placed as un-archived folders while for HDFS these need to be archived files in `.zip` format. |

| Parameter | Description |
|---|---|
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |

## Output Columns

1. AdditionalInputData
2. AddressLine1-4
3. City
4. Country
5. FirmName
6. PostalCode
7. PostalCode.AddOn
8. PostalCode.Base
9. StateProvince
10. AddressBlock1-9
11. ApartmentLabel

12. ApartmentNumber
13. ApartmentNumber2
14. Building
15. City
16. Country
17. County *
18. FirmName
19. HouseNumber
20. LeadingDirectional
21. POBox
22. PostalCode
23. Principality *
24. StateProvince
25. StreetAlias
26. StreetName
27. StreetSuffix
28. Subcity *
29. Substreet *
30. TrailingDirectional
31. ApartmentLabel.Input
32. ApartmentNumber.Input
33. City.Input
34. Country.Input
35. County.Input *
36. FirmName.Input
37. HouseNumber.Input
38. LeadingDirectional.Input
39. POBox.Input
40. PostalCode.Input
41. Principality.Input *
42. StateProvince.Input
43. StreetAlias.Input
44. StreetName.Input
45. StreetSuffix.Input
46. Subcity.Input *
47. Substreet.Input *
48. TrailingDirectional.Input
49. Geocode.MatchCode
50. Latitude
51. Longitude
52. SearchDistance

53. Confidence
54. CouldNotValidate
55. MatchScore
56. ProcessedBy
57. Status
58. Status.Code
59. Status.Description
60. ApartmentLabel.Result
61. ApartmentNumber.Result
62. City.Result
63. Country.Result
64. County.Result *
65. FirmName.Result
66. HouseNumber.Result
67. LeadingDirectional.Result
68. POBox.Result
69. PostalCode.Result
70. PostalCode.Type
71. Principality.Result *
72. StateProvince.Result
73. StreetAlias.Result
74. StreetName.Result
75. StreetSuffix.Result
76. Subcity.Result *
77. Substreet.Result *
78. TrailingDirectional.Result
79. Barcode
80. DPID
81. FloorNumber
82. FloorType
83. PostalBoxNum

*This is a subfield and may not contain data.

**Table 2: City/Street/Postal Code Centroid Match Codes**

| Element | Match Code |
|---|---|
| Address Point | P4 |
| Address Point Interpolated | I4 |

| Element | Match Code |
|---|---|
| Street Centroid | A4/P3 |
| Postal Code/City Centroid | A3/P2/A2 |

**Note:** For the field descriptions, see the *Validate Address Loqate* topic of the *Addressing Guide* of Spectrum™ Technology Platform.

### Using a Validate Address Loqate MapReduce Job

1. Create an instance of `LoqateAddressingFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Validate Address Loqate job by creating an instance of `LoqateAddressingDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32. For this, the steps are:
   a) Configure the JVM initialization settings by creating an instance of `LoqateAddressingGeneralConfiguration`.

      Use the enum **Enum ExhaustedAction** on page 362.
   b) Configure the necessary database settings by creating an instance of `LoqateAddressingEngineConfiguration` and set the various fields.
   c) Configure the address validation settings by creating an instance of `LoqateAddressingValidateConfiguration`.

      To set the values of the various fields of this instance, use the enums **Enum AcceptanceLevel** on page 363, **Enum CountryCodes** on page 360, **Enum OutputCasing** on page 363, **Enum CountryFormat** on page 363, and **Enum ScriptAlphabet** on page 363.
   d) Set the details of the Reference Data path by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.
   e) Create an instance of `LoqateAddressingDetail`, by passing an instance of type `JobConfig`, the `LocalReferenceDataPath` instance, and the `LoqateAddressingValidateConfiguration` instance created earlier as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.

      1. Set the details of the input file using the `inputPath` field of the `LoqateAddressingDetail` instance.

         **Note:**

         - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
         - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.

- For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

2. Set the details of the output file using the `outputPath` field of the `LoqateAddressingDetail` instance.

   **Note:**

   - For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
   - For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
   - For a parquet output file, create an instance of `ParquetFilePath` with the path of the parquet output file as the argument.

3. Set the name of the job using the `jobName` field of the `LoqateAddressingDetail` instance.

3. To create a MapReduce job, use the previously created instance of `LoqateAddressingFactory` to invoke its method `createJob()`. In this, pass the above instance of `LoqateAddressingDetail` as an argument.
   The `createJob()` method returns a `List` of instances of `ControlledJob`.
4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters post a successful MapReduce job run, use the previously created instance of `LoqateAddressingFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using a Validate Address Loqate Spark Job

1. Create an instance of `LoqateAddressingFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Validate Address Loqate job by creating an instance of `LoqateAddressingDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32. For this, the steps are:
   a) Configure the JVM initialization settings by creating an instance of `LoqateAddressingGeneralConfiguration`.
      Use the enum **Enum ExhaustedAction** on page 362.
   b) Configure the necessary database settings by creating an instance of `LoqateAddressingEngineConfiguration` and set the various fields.
   c) Configure the address validation settings by creating an instance of `LoqateAddressingValidateConfiguration`.
      To set the values of the various fields of this instance, use the enums **Enum AcceptanceLevel** on page 363, **Enum CountryCodes** on page 360, **Enum OutputCasing** on page 363, **Enum CountryFormat** on page 363, and **Enum ScriptAlphabet** on page 363.

d) Set the details of the Reference Data path by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.

e) Create an instance of `LoqateAddressingDetail`, by passing an instance of type `JobConfig`, the `LocalReferenceDataPath` instance, and the `LoqateAddressingValidateConfiguration` instance created earlier as the arguments to its constructor.

The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.

1. Set the details of the input file using the `inputPath` field of the `LoqateAddressingDetail` instance.

**Note:**

- For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
- For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
- For a parquet input file, create an instance of `ParquetFilePath` with the path of the parquet input file as the argument.

2. Set the details of the output file using the `outputPath` field of the `LoqateAddressingDetail` instance.

**Note:**

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a parquet output file, create an instance of `ParquetFilePath` with the path of the parquet output file as the argument.

3. Set the name of the job using the `jobName` field of the `LoqateAddressingDetail` instance.

3. To create and run the Spark job, use the previously created instance of `LoqateAddressingFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `LoqateAddressingDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. Display the counters to view the reporting statistics for the job.

# Universal Name Module Jobs

## Common Module API

### UniversalNameDetail<T extends ProcessType>

*Purpose*
To specify the details of a Universal Name Module job.

### UniversalNameFactory

*Purpose*
A singleton factory class to create instances of Universal Name Module jobs.

## Open Name Parser

Open Name Parser breaks down personal and business names and other terms in the name data field into their component parts. These parsed name elements are then subsequently available to other automated operations such as name matching, name standardization, or multi-record name consolidation.

### API Entities

#### *OpenNameParserDetail*

*Purpose*
To specify details of an Open Name Parser job.

#### *OpenNameParserConfiguration*

*Purpose*
To break down personal and business names and other terms in the `name` data field into their component parts.

## Input Parameters

| Parameter | Description |
|---|---|
| Open Name Parser Configuration | To break down personal and business names and other terms in the `name` data field into their component parts. |
| Reference Data Path | To specify the Reference Data path details. |
| Job Configurations | The Hadoop configurations for the job.<br>For a MapReduce job, the instance must be of type **MRJobConfig** on page 30. For a Spark job, the instance must be of type **SparkJobConfig** on page 31. |

| Parameter | Description |
|---|---|
| Input File | *For text files:*<br>**File Path**<br>The path of the input text file on the Hadoop platform.<br>**Record Separator**<br>The record separator used in the input file.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the input file.<br>**Text Qualifier**<br>The character used to surround text values in a delimited file.<br>**Header Row Fields**<br>An array of the header fields of the input file.<br>**Skip First Row**<br>Flag to indicate if the first row must be skipped while reading the input file records.<br>This must be `true` in case the first row is a header row.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the input ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the input Parquet format file on the Hadoop platform.<br>*Common parameters:*<br>**Field Mappings**<br>A map of key value pairs, with the existing column names as the keys and the desired output column names as the values. |

| Parameter | Description |
|---|---|
| Output File | *For text files:*<br>**File Path**<br>The path of the output text file on the Hadoop platform.<br>**Field Separator**<br>The separator used between any two consecutive fields of a record, in the output file.<br>**Attention:** Invoke the appropriate constructor of `FilePath`.<br>*For ORC format files:*<br>**ORC File Path**<br>The path of the output ORC format file on the Hadoop platform.<br>*For Parquet format files:*<br>**Parquet File Path**<br>The path of the output Parquet format file on the Hadoop platform.<br>*Common Parameters:*<br>**Overwrite**<br>Flag to indicate if output file must overwrite any existing file of same name.<br>**Create Output Header**<br>Flag to indicate if header file is to be created on the Hadoop server or not. |
| Job Name | The name of the job. |

## Output Columns

In addition to the input columns, the following columns are added while generating the output of an Open Name Parser job:

| | Format | Description |
|---|---|---|
| AccountDescription | String | An account description that is part of the name. For example, in "Mary Jones Account # 12345", the account description is "Account#12345". |
| **Fields Related to Names of Companies** | | |

| | Format | Description |
|---|---|---|
| FirmConjunction | String | Indicates that the name of a firm contains a conjunction such as "d/b/a" (doing business as), "o/a" (operating as), and "t/a" (trading as). |
| FirmName | String | The name of a company. For example, "Pitney Bowes". |
| FirmSuffix | String | The corporate suffix. For example, "Co." and "Inc." |
| IsFirm | String | Indicates that the name is a firm rather than an individual. Values are true or false. |
| **Fields Related to Names of Individual People** | | |
| Conjunction | String | Indicates that the name contains a conjunction such as "and", "or", or "&". |
| CultureCode | String | The culture codes contained in the input data. |
| CultureCodeUsedToParse | String | Identifies the culture-specific grammar that was used to parse the data.<br>**Null (empty)** Global culture (default).<br>**de** German.<br>**es** Spanish.<br>**ja** Japanese. |
| FirstName | String | The first name of a person. |
| GeneralSuffix | String | A person's general/professional suffix. For example, MD or PhD. |
| IsParsed | String | Indicates whether an output record was parsed. Values are true or false. |

| | Format | Description |
|---|---|---|
| IsPersonal | String | Indicates whether the name is an individual rather than a firm. Values are true or false. |
| IsReverseOrder | String | Indicates whether the input name is in reverse order. Values are true or false. |
| LastName | String | The last name of a person. Includes the paternal last name. |
| LeadingData | String | Non-name information that appears before a name. |
| MaturitySuffix | String | A person's maturity/generational suffix. For example, Jr. or Sr. |
| MiddleName | String | The middle name of a person. |
| Name. | String | The personal or firm name that was provided in the input. |
| NameScore | String | Indicates the average score of known and unknown tokens for each name. The value of NameScore will be between 0 and 100, as defined in the parsing grammar. 0 is returned when no matches are returned. |
| SecondaryLastName | String | In Spanish parsing grammar, the surname of a person's mother. |
| TitleOfRespect | String | Information that appears before a name, such as "Mr.", "Mrs.", or "Dr." |
| TrailingData | String | Non-name information that appears after a name. |
| **Fields Related to Conjoined Names** | | |
| Conjunction2 | String | Indicates that a second, conjoined name contains a conjunction such as "and", "or", or "&". |

| | Format | Description |
|---|---|---|
| Conjunction3 | String | Indicates that a third, conjoined name contains a conjunction such as "and", "or", or "&". |
| FirmName2 | String | The name of a second, conjoined company. For example, Baltimore Gas & Electric dba Constellation Energy. |
| FirmSuffix2 | String | The suffix of a second, conjoined company. |
| FirstName2 | String | The first name of a second, conjoined name. |
| FirstName3 | String | The first name of a third, conjoined name. |
| GeneralSuffix2 | String | The general/professional suffix for a second, conjoined name. For example, MD or PhD. |
| GeneralSuffix3 | String | The general/professional suffix for a third, conjoined name. For example, MD or PhD. |
| IsConjoined | String | Indicates that the input name is conjoined. An example of a conjoined name is "John and Jane Smith." Values are true or false. |
| LastName2 | String | The last name of a second, conjoined name. |
| LastName3 | String | The last name of a third, conjoined name. |
| MaturitySuffix2 | String | The maturity/generational suffix for a second, conjoined name. For example, Jr. or Sr. |
| MaturitySuffix3 | String | The maturity/generational suffix for a third, conjoined name. For example, Jr. or Sr. |

| | Format | Description |
|---|---|---|
| MiddleName2 | String | The middle name of a second, conjoined name. |
| MiddleName3 | String | The middle name of a third, conjoined name. |
| TitleOfRespect2 | String | Information that appears before a second, conjoined name, such as "Mr.", "Mrs.", or "Dr." |
| TitleOfRespect3 | String | Information that appears before a third, conjoined name, such as "Mr.", "Mrs.", or "Dr." |

### Using an Open Name Parser MapReduce Job

1. Create an instance of `UniversalNameFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Open Name Parser job by creating an instance of `OpenNameParserDetail` specifying the `ProcessType`. The instance must use the type **MRProcessType** on page 32.
   a) Configure the open name parser rules by creating an instance of `OpenNameParserConfiguration`.
   b) Set the details of the Reference Data path and location type by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.
   c) Create an instance of `OpenNameParserDetail`, by passing an instance of type `JobConfig`, and the `OpenNameParserConfiguration` and `ReferenceDataPath` instances created earlier as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 30.
   d) Set the details of the input file using the `inputPath` field of the `OpenNameParserDetail` instance.

      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
      - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
      - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.

   e) Set the details of the output file using the `outputPath` field of the `OpenNameParserDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `OpenNameParserDetail` instance.

3. To create a MapReduce job, use the previously created instance of `UniversalNameFactory` to invoke its method `createJob()`. In this, pass the above instance of `OpenNameParserDetail` as an argument.
   The `createJob()` method returns a `List` of instances of `ControlledJob`.
4. Run the created job using an instance of `JobControl`.
5. To display the reporting counters post a successful MapReduce job run, use the previously created instance of `UniversalNameFactory` to invoke its method `getCounters()`, passing the created job as an argument.

### Using an Open Name Parser Spark Job

1. Create an instance of `UniversalNameFactory`, using its static method `getInstance()`.
2. Provide the input and output details for the Open Name Parser job by creating an instance of `OpenNameParserDetail` specifying the `ProcessType`. The instance must use the type **SparkProcessType** on page 32.
   a) Configure the open name parser rules by creating an instance of `OpenNameParserConfiguration`.
   b) Set the details of the Reference Data path and location type by creating an instance of `ReferenceDataPath`. See **Enum ReferenceDataPathLocation** on page 357.
   c) Create an instance of `OpenNameParserDetail`, by passing an instance of type `JobConfig`, and the `OpenNameParserConfiguration` and `ReferenceDataPath` instances created earlier as the arguments to its constructor.
      The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 31.
   d) Set the details of the input file using the `inputPath` field of the `OpenNameParserDetail` instance.
      - For a text input file, create an instance of `FilePath` with the relevant details of the input file by invoking the appropriate constructor.
      - For an ORC input file, create an instance of `OrcFilePath` with the path of the ORC input file as the argument.
      - For a Parquet input file, create an instance of `ParquetFilePath` with the path of the Parquet input file as the argument.
   e) Set the details of the output file using the `outputPath` field of the `OpenNameParserDetail` instance.

- For a text output file, create an instance of `FilePath` with the relevant details of the output file by invoking the appropriate constructor.
- For an ORC output file, create an instance of `OrcFilePath` with the path of the ORC output file as the argument.
- For a Parquet output file, create an instance of `ParquetFilePath` with the path of the Parquet output file as the argument.

f) Set the name of the job using the `jobName` field of the `OpenNameParserDetail` instance.

3. To create and run the Spark job, use the previously created instance of `UniversalNameFactory` to invoke its method `runSparkJob()`. In this, pass the above instance of `OpenNameParserDetail` as an argument.

   The `runSparkJob()` method runs the job and returns a `Map` of the reporting counters of the job.

4. Display the counters to view the reporting statistics for the job.

# 5 - XML Configuration Files

## In this section

# Sample Configuration Files

The sample configuration XML files provide a simpler way of running the map reduce and spark jobs for various address and data quality activities. These files are targeted for users who want to run the jobs without having to understand java code. The files contain properties in the form of key-value pairs, which can be modified according to need.

You can run the required job using command prompt (for Linux system) and an SSH client, such as Putty (for Windows and Unix systems).

The sample configuration files are shipped as part of the Spectrum™ Technology Platform SDK and you can access these at this location after you install the SDK:

- `<Big Data Quality bundle>\samples\configuration\mr`: For MR jobs
- `<Big Data Quality bundle>\samples\configuration\spark`: For Spark jobs

### *File Types*

Each of the folders on these locations has these types of configuration XML files, which have properties, in the form of parameters and values, needed to process the jobs. You can customize the values according to the requirement of the job you are running.

- *inputFileConfig.xml*: Specifies the input file properties, such as the type of the input file, path where it's kept, record delimiters, field delimiters, text qualifiers, and the file header details.
- *<job>Config.xml* (for example, *addressValidationConfig*): Specifies job-related properties, such as job type, job name, input options, and rule configuration or engine configuration.
- *mapReduceConfig.xml*: Specifies the MapReduce configuration parameters. Use this file for customizing any of the MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job.
- *OutputFileConfig.xml*: Specifies the type of output file, its location, field delimiter used in the file, if the header file needs to be created, and if report counters are to be printed to file or on console.

## Using Configuration Property Files

Ensure the Spectrum™ Data & Address Quality for Big Data SDK is installed on your machine.

You can run a Spectrum™ Data & Address Quality for Big Data SDK job using the module-specific JAR files and the configuration files in XML formats.

For a list of the module-specific JAR files, see **Components of the SDK Java API** on page 26.

1. For a Linux system, open a command prompt.

   For Windows and Unix systems, open an SSH client, such as *Putty*.

2. For a *MapReduce* job, use the command `hadoop`.

   Based on the job you wish to run:

   1. Pass the name of the JAR file of that module.
   2. Pass the driver class's name `RunMRSampleJob`.
   3. Pass the various configuration files as a list of arguments. Each argument key accepts the path of a single configuration property file, where each file contains multiple configuration properties.

   The syntax of the command is:

   ```
   hadoop jar <Name of module JAR file> RunMRSampleJob [-config <Path to
   configuration file>] [-debug] [-input <Path to input configuration
   file>] [-conf <Path to MapReduce configuration file>] [-output <Path
   of output directory>]
   ```

   For example, for a MapReduce MatchKeyGenerator job:

   ```
   hadoop jar amm.core.12.2.jar RunMRSampleJob -config
   /home/hadoop/matchkey/mkgConfig.xml -input
   /home/hadoop/matchkey/inputFileConfig.xml -conf
   /home/hadoop/matchkey/mapReduceConfig.xml -output
   /home/hadoop/matchkey/outputFileConfig.xml
   ```

3. For a *Spark* job, use the command `spark-submit`.

   Based on the job you wish to run:

   1. Pass the name of the JAR file of that module.
   2. Pass the driver class's name `RunSparkSampleJob`.
   3. Pass the various configuration files as a list of arguments. Each argument key accepts the path of a single configuration property file, where each file contains multiple configuration properties.

   The syntax of the command is:

   ```
   spark-submit --class RunSparkSampleJob <Name of module JAR file>
   [-config <Path to configuration file>] [-debug] [-input <Path to input
   configuration file>] [-conf <Path to Spark configuration file>] [-output
   <Path of output directory>]
   ```

   For example, for a Spark MatchKeyGenerator job:

   ```
   spark-submit --class RunSparkSampleJob amm.core.12.2.jar -config
   /home/hadoop/spark/matchkey/matchKeyGeneratorConfig.xml -input
   /home/hadoop/spark/matchkey/inputFileConfig.xml -output
   /home/hadoop/spark/matchkey/outputFileConfig.xml
   ```

**Note:** To see a list of argument keys supported for the `hadoop` or `spark-submit` commands, run the commands:

```
hadoop --help
```

or

```
spark-submit --help
```

# Advanced Matching Module

## Best of Breed

Best of Breed consolidates duplicate records by selecting the best data in a duplicate record collection and creating a new consolidated record using the best data. This "super" record is known as the best of breed record. You define the rules to use in selecting records to process. When processing completes, the best of breed record is retained by the system.

### Configuration Files

These tables describe the parameters and the values you need to specify before you run the Best of Breed job.

**Table 3: inputFileConfig**

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/bestofbreed/input/BestOfBreed_Input.csv` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |

| Parameter | Description |
|---|---|
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

**Table 4: bestOfBreedConfig**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `BestOfBreed`. |
| *pb.bdq.job.name* | Name of the job. Default is `BestOfBreedSample`. |
| *pb.bdq.match.groupby* | Name of the column to be used for grouping records in the match queue. |
| *pb.bdq.reduce.count* | Number of reducers to be run. Default is `1`. |
| *pb.bdq.consolidation.json* | Json string to define the rules for consolidating duplicate records. |

**Table 5: mapReduceConfig**

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

**Table 6: OutputFileConfig**

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/bestofbreed/output`. |

| Parameter | Description |
|---|---|
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |

| Parameter | Description |
|---|---|
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Duplicate Synchronization

Duplicate Synchronization determines which fields from a collection of records to copy to the corresponding fields of all records in the collection. You can specify the rules that records must satisfy in order to copy the field data to the other records in the collection. When processing has been completed, all records in the collection are retained.

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Duplicate Synchronization job.

### Table 7: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/duplicatesync/input/DuplicateSync_Input.csv` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 8: duplicateSyncConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `DuplicateSynchronization`. |
| *pb.bdq.job.name* | Name of the job. Default is `DuplicateSynchronizationSample`. |
| *pb.bdq.match.groupby* | Name of the column to be used for grouping records in the match queue. |
| *pb.bdq.reduce.count* | Number of reducers to be run. Default is `1`. |
| *pb.bdq.consolidation.json* | Json string for duplicate synchronization consolidation rules. |

### Table 9: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 10: OutputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `user/hduser/sampledata/duplicatesync/output`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |

| Parameter | Description |
|---|---|
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Filter

The Filter stage retains or removes records from a group of records based on the rules you specify.

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Filter job.

### Table 11: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/filter/input/Filter_Input.csv` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 12: filterConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `Filter`. |
| *pb.bdq.job.name* | Name of the job. Default is `FilterSample`. |

| Parameter | Description |
|---|---|
| *pb.bdq.match.groupby* | Name of the column to be used for grouping records in the match queue. |
| *pb.bdq.reduce.count* | Number of reducers to be run. Default is `1`. |
| *pb.bdq.consolidation.json* | Json string for duplicate synchronization consolidation rules. |

### Table 13: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 14: OutputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `user/hduser/sampledata/filter/output`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |

| Parameter | Description |
|---|---|
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Interflow Match

Interflow Match locates matches between similar data records across two input record streams. The first record stream is a source for suspect records and the second stream is a source for candidate records.

Using match group criteria (for example a match key), Interflow Match identifies a group of records that are potentially duplicates of a particular suspect record.

Each candidate is separately matched to the Suspect and is scored according to your match rules. If the candidate is a duplicate, it is assigned a collection number, the match record type is labeled a duplicate, and written out; unmatched unique candidates may be written out at the user's option. When Interflow Match has exhausted all candidate records in the current match group, the matched suspect record is assigned a collection number that corresponds to its duplicate record. Or, if no matches where identified, the suspect is assigned a collection number of 0 and is labeled a unique record.

> **Note:** Interflow Match only matches suspect records to candidate records. It does not attempt to match suspect records to other suspect records as is done in Intraflow Match.

The matching process for a particular suspect may terminate before matching all possible candidates if you have set a limiter on duplicates and the limit has been exceeded for the current suspect.

The type of matching (Intraflow or Interflow) determines how express key match results translate to Candidate Match Scores. In Interflow matching, a successful Express Key match always confers a 100 MatchScore onto the Candidate. On the other hand, in Intraflow matching, the score a Candidate gains as a result of an Express Key match depends on whether the record to which that Candidate matched was a match of some other Suspect—Express Key duplicates of a Suspect will always have MatchScores of 100, whereas Express Key duplicates of another Candidate (which was a duplicate of a Suspect) will inherit the MatchScore (not necessarily 100) of that Candidate

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Interflow Match job.

### Table 15: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| **Suspect File** | |
| *pb.bdq.match.suspect.inputfile.path* | Path where you have placed the suspect input file on HDFS. Example: `/user/hduser/sampledata/intermatch/input/Interflow_Suspect.txt`. |
| *pb.bdq.match.suspect.recordseparator* | Record delimiter used in the suspect file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.match.suspect.fieldseparator* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |

| Parameter | Description |
|---|---|
| *pb.bdq.match.suspect.textqualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.match.suspect.header* | Headers used in the suspect file. Example: `name`, `firstname`, `lastname`, `matchkey`, `middlename`, and `recordid`. |
| *pb.bdq.match.suspect.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |
| **Candidate File** | |
| *pb.bdq.match.candidate.inputfile.path* | Path where you have placed the candidate input file on HDFS. Example: `/user/hduser/sampledata/intermatch/input/Interflow_candidate.txt`. |
| *pb.bdq.match.candidate.recordseparator* | Record delimiter used in the candidate file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.match.candidate.fieldseparator* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.match.candidate.textqualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.match.candidate.header* | Headers used in the candidate file. Example: `name`, `firstname`, `lastname`, `matchkey`, `middlename`, and `recordid`. |
| *pb.bdq.match.candidate.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 16: interMatchConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `InterMatch`. |
| *pb.bdq.job.name* | Name of the job. Default is `InterMatchSample`. |

| Parameter | Description |
|---|---|
| *pb.bdq.match.rule* | Json String for defining match rule. It specifies details, such as match rule hierarchy, matching method, method to score blank data in a field, scoring method, and algorithm to determine if the values in the field name matched. |
| *pb.bdq.match.groupby* | Name of the column to be used for grouping records in the match queue. |
| *pb.bdq.reduce.count* | Number of reducers to be run. Default is `1`. |
| *pb.bdq.match.express.column* | Name of the Express Match Column. If the content of this column matches between the suspect and the candidate, no further processing is needed to determine if the suspect and the candidates are duplicates. |
| *pb.bdq.match.keygenerator.json* | Json string for defining match key generator rule, such as whether to use expressMatchKey, name of the matchKeyField, and algorithm to be used.<br>**Note:** This is an optional detail. |
| *pb.bdq.match.unique.collectnumber.zero* | A `true` value assigns collection number `0` to unique records. |
| *pb.bdq.match.inter.comparison* | Inter match comparison options.<br>• returnUniqueCandidates: Set the value to `true` to return unique records within a match group.<br>• maxNumOfDuplicates: Specify the maximum number of duplicates to be found before stopping the comparison in the match group.<br>**Note:** This is an optional detail. |

### Table 17: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

**Table 18: Output File Configuration**

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output to be generated on HDFS. For example, `/user/hduser/sampledata/intermatch/output`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| *pb.bdq.job.print.counters.console* | If the counters are printed on console or in a file. `True` indicates counters are printed on the console |
| *pb.bdq.job.counter.file.path* | Path and the name of the file to which the counters are to be printed. You need to specify this if value in the *pb.bdq.job.print.counters.console* is `false`. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |

| Parameter | Description |
|---|---|
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Intraflow Match

Intraflow Match locates matches between similar data records within a single input stream. You can create hierarchical rules based on any fields that have been defined or created in other stages of the dataflow.

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Intraflow Match job.

### Table 19: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/intramatch/input/Intraflow_Input.txt`. |

| Parameter | Description |
|---|---|
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Headers used in the input file. Example: `name`, `firstname`, `lastname`, `matchkey`, `middlename`, and `recordid`. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 20: intraMatchConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `IntraMatch`. |
| *pb.bdq.job.name* | Name of the job. Default is `IntraMatchSample`. |
| *pb.bdq.match.rule* | Json String for defining match rule. It specifies details, such as match rule hierarchy, matching method, method to score blank data in a field, scoring method, and algorithm to determine if the values in the field name matched. |
| *pb.bdq.match.groupby* | Name of the column to be used for grouping records in the match queue. |
| *pb.bdq.reduce.count* | Number of reducers to be run. Default is `1`. |
| *pb.bdq.match.express.column* | Name of the Express Match Column. If the content of this column matches between the suspect and the candidate, no further processing is needed to determine if the suspect and the candidates are duplicates. |
| *pb.bdq.match.keygenerator.json* | Json string for defining match key generator rule, such as whether to use expressMatchKey, name of the matchKeyField, and algorithm to be used.<br>**Note:** This is an optional detail. |

| Parameter | Description |
|---|---|
| *pb.bdq.match.unique.collectnumber.zero* | A `true` value assigns collection number `0` to unique records. |

### Table 21: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 22: Output File Configuration

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/intramatch/output`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| *pb.bdq.job.print.counters.console* | If the counters are printed on console or in a file. `True` indicates counters are printed on the console |
| *pb.bdq.job.counter.file.path* | Path and the name of the file to which the counters are to be printed. You need to specify this if value in the *pb.bdq.job.print.counters.console* is `false`. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |

| Parameter | Description |
|---|---|
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding.<br>Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`.<br>Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

## Match Key Generator

Match Key Generator creates a non-unique key for each record, which can then be used by matching stages to identify groups of potentially duplicate records. Match keys facilitate the matching process by allowing you to group records by match key and then only comparing records within these groups.

The match key is created using rules you define and is comprised of input fields. Each input field specified has a selected algorithm that is performed on it. The result of each algorithm is then concatenated to create a single match key field.

In addition to creating match keys, you can also create express match keys to be used later in the dataflow by an Intraflow Match stage or an Interflow Match stage.

You can create multiple match keys and express match keys.

For example, if the incoming record is:

First Name - Fred
Last Name - Mertz
Postal Code - 21114-1687
Gender Code - M

And you define a match key rule that generates a match key by combining data from the record like this:

| Input Field | Start Position | Length |
|---|---|---|
| Postal Code | 1 | 5 |
| Postal Code | 7 | 4 |
| Last Name | 1 | 5 |
| First Name | 1 | 5 |
| Gender Code | 1 | 1 |

Then the key would be:

211141687MertzFredM

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Match Key Generator job.

### Table 23: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |

| Parameter | Description |
|---|---|
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/matchkeygenerator/input/MatchKey_Input.csv`. |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Column headers as comma-separated values. For example, businessname, id, and domain. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 24: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 25: matchKeyGeneratorConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `MatchKeyGen`. |
| *pb.bdq.job.name* | Name of the job. Default is `MatchKeySample`. |
| *pb.bdq.match.keygenerator.json* | Json string for match key generator rules, such as algorithm to be used to generate the match key, field to which you want to apply the selected algorithm, starting position within the specified field, length of characters to include from the starting position, if non-numeric and non-alpha characters are to be removed, and if the input fields are to be sorted. |

### Table 26: outputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Output file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding.<br>Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`.<br>Default is `PARQUET_1_0`. |

| Parameter | Description |
|---|---|
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Transactional Match

Transactional Match matches suspect records against candidate records that are returned from the Candidate Finder stage. Transactional Match uses matching rules to compare the suspect record to all candidate records with the same candidate group number (assigned in Candidate Finder) to identify duplicates. If the candidate record is a duplicate, it is assigned a collection number, the match record type is labeled a Duplicate, and the record is then written out. Any unmatched candidates in the group are assigned a collection number of 0, labeled as Unique and then written out as well.

> **Note:** Transactional Match only matches suspect records to candidates. It does not attempt to match suspect records to other suspect records as is done in Intraflow Match.

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Transactional Match job.

### Table 27: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/transactionalmatch/input/TransMatch_Input.txt` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |

| Parameter | Description |
|---|---|
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Headers used in the input file. Example: `name`, `firstname`, `lastname`, `CandidateGroup`, `middlename`, and `recordid`. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 28: transactionalMatchConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `Transactional`. |
| *pb.bdq.job.name* | Name of the job. Default is `TransactionalMatchSample`. |
| *pb.bdq.match.rule* | Json String for defining match rule. It specifies details, such as match rule hierarchy, matching method, method to score blank data in a field, scoring method, and algorithm to determine if the values in the field name matched. |
| *pb.bdq.match.groupby* | Name of the column to be used for grouping records in the match queue. |
| *pb.bdq.reduce.count* | Number of reducers to be run. Default is `1`. |
| *pb.bdq.match.keygenerator.json* | Json string to indicate if an expressMatchKey is to be created, specify matchKeyField, and the rules for generating match key.<br>**Note:** Optional detail |
| *pb.bdq.match.unique.candidate.return* | Set to `true` if you want unique candidate records to be included in the output. |

### Table 29: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 30: OutputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/transactionalmatch/output`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| *pb.bdq.job.print.counters.console* | If the counters are printed on console or in a file. `True` indicates counters are printed on the console |
| *pb.bdq.job.counter.file.path* | Path and the name of the file to which the counters are to be printed. You need to specify this if value in the *pb.bdq.job.print.counters.console* is `false`. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |

| Parameter | Description |
|---|---|
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding.<br>Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`.<br>Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Data Integration Module

## Custom Groovy Script

### Configuration Files

These tables describe the parameters and values you need to specify before you run the Custom Groovy Script job.

### Table 31: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/groovy/input/groovy_Input.csv` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 32: scriptExecuterConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `CustomScript`. |
| *pb.bdq.job.name* | Name of the job. Default is `CustomScriptSample`. |
| *pb.bdq.dim.date.pattern* | Specifies the date pattern to be used in the job as: *M/d/yy*<br>**Note:** This is an optional property. |
| *pb.bdq.dim.datetime.pattern* | Specifies the date-time pattern to be used in the job as: *M/d/yy h:mm a*<br>**Note:** This is an optional property. |
| *pb.bdq.dim.time.pattern* | Specifies the time pattern to be used in the job as: *h:mm a*<br>**Note:** This is an optional property. |

| Parameter | Description |
|---|---|
| *pb.bdq.dim.groovy.input.fields.0* | Specifies the input fields and their data types in the format: `{"name":<"name of the field">,"type":<"field type">}`.<br>For example, `{"name":"AddressLine2","type":"string"}` |
| *pb.bdq.dim.groovy.output.fields.0* | Specifies the input fields and their data types in the format: `{"name":<"name of the field">,"type":<"field type">}`.<br>For example, `{"name":"AddressLine2","type":"string"}` |
| *pb.bdq.dim.groovy.script.0* | Path of the groovy script to be executed. For Example, `/home/hduser/script/groovy.txt` |

### Table 33: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 34: OutputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/groovy/output` |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| **Properties of Parquet file** | |

| Parameter | Description |
|---|---|
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Joiner

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Joiner job.

> **Note:** The description here assumes there are three input files for the joiner job. However, you can have any number of input files for the job.

### Table 35: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| **These rows describe the details of the first input file.** | |
| *pb.bdq.inputfile.path.0* | The path where you have placed the input file on HDFS. For example, `/home/hduser/input/input0.txt` |
| *textinputformat.record.delimiter.0* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter.0* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier.0* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header.0* | Column headers as comma-separated values. For example, business name, id, and domain. |
| *pb.bdq.inputformat.skip.firstrow.0* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |
| **These rows describe the details of the second input file.** | |
| *pb.bdq.inputfile.path.1* | The path where you have placed the input file on HDFS. For example, `/home/hduser/input/input1.txt` |

| Parameter | Description |
|---|---|
| *textinputformat.record.delimiter.1* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter.1* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier.1* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header.1* | Column headers as comma-separated values. For example, business name, id, and domain. |
| *pb.bdq.inputformat.skip.firstrow.1* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |
| **These rows describe details of the third input file.** | |
| *pb.bdq.inputfile.path.2* | The path where you have placed the input file on HDFS. For example, `/home/hduser/input/input2.txt` |
| *textinputformat.record.delimiter.2* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter.2* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier.2* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header.2* | Column headers as comma-separated values. For example, business name, id, and domain. |
| *pb.bdq.inputformat.skip.firstrow.2* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 36: joinerConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `Joiner`. |
| *pb.bdq.job.name* | Name of the job. Default is `JoinerSample`. |

| Parameter | Description |
|---|---|
| *com.pb.bdq.dim.join.left.port* | Json string for defining input File Index of left port. |
| *com.pb.bdq.dim.join.type* | Specify the type of join operation to be performed. Options are:<br>• `LeftOuter`<br>• `Full`<br>• `Inner` |
| *com.pb.bdq.dim.join.col.0* | Specify the columns to be joined, in comma separated format (,). |
| *com.pb.bdq.dim.join.col.1* | Specify the columns to be joined, in comma separated format (,). |
| *com.pb.bdq.dim.join.col.2* | Specify the columns to be joined, in comma separated format (,). |

### Table 37: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 38: OutputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/joiner/output` |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |

| Parameter | Description |
|---|---|
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Data Normalization Module

## Advanced Transformer

The Advanced Transformer job scans and splits strings of data into multiple fields using tables or regular expressions. It extracts a specific term or a specified number of words to the right or left of a term. Extracted and non-extracted data can be placed into an existing field or a new field.

For example, want to extract the suite information from this address field and place it in a separate field.

2300 BIRCH RD STE 100

To accomplish this, you could create an Advanced Transformer that extracts the term STE and all words to the right of the term STE, leaving the field as:

2300 BIRCH RD

### Configuration Files

These tables describe the parameters and the values you need to specify before you run the Advanced Transformer job.

**Table 39: inputFileConfig**

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/advancedtransformer/input/AdvancedTransformer_Input.txt` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |

| Parameter | Description |
|---|---|
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |
| *pb.bdq.inputfile.field.mapping* | Maps the values of the headers used in the input file with their updated values.<br>**Note:** This is an optional parameter. |

**Table 40: advancedTransformerConfig**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `AdvTransformer`. |
| *pb.bdq.job.name* | Name of the job. Default is `AdvanceTransformerSample`. |
| *pb.bdq.dnm.advtransformer.configuration* | Json string for defining advance transformer configuration. It specifies details, such as the source input field to be evaluated for scan and split, the output field where you want to put the extracted data, any special characters that you want to tokenize, and the type of extraction to be performed. |
| *pb.bdq.reference.data* | The path where you have placed the reference data. For example, `{"referenceDataPathLocation": "LocaltoDataNodes","dataDir":" /home/data/referenceData"}` |

**Table 41: advancedTransformerConfigHDFSRefData(DataDownloader)**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `AdvTransformer`. |
| *pb.bdq.job.name* | Name of the job. Default is `AdvanceTransformerSample`. |

| Parameter | Description |
|---|---|
| *pb.bdq.dnm.advtransformer.configuration* | Json string for defining advance transformer configuration. It specifies details, such as the source input field to be evaluated for scan and split, the output field where you want to put the extracted data, any special characters that you want to tokenize, and the type of extraction to be performed. |
| *pb.bdq.reference.data* | Path of reference data on HDFS and the data downloader path. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/home/data/dm/referenceData", "dataDownloader":{"dataDownloader":"HDFS", "localFSRepository":"/local/download"}}` |

**Table 42: advancedTransformerConfigDistributedCache**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `AdvTransformer`. |
| *pb.bdq.job.name* | Name of the job. Default is `AdvanceTransformerSample`. |
| *pb.bdq.dnm.advtransformer.configuration* | Json string for defining advance transformer configuration. It specifies details, such as the source input field to be evaluated for scan and split, the output field where you want to put the extracted data, any special characters that you want to tokenize, and the type of extraction to be performed. |
| *pb.bdq.reference.data* | Path of the reference data on HDFS and the type of data downloader. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/home/data/dm/referenceData", "dataDownloader":{"dataDownloader":"DC"}}` |

**Table 43: mapReduceConfig**

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 44: OutputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/advancedtransformer/output` |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |

| Parameter | Description |
|---|---|
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Open Parser

Open Parser parses your input data from many cultures of the world using a simple but powerful parsing grammar. Using this grammar, you can define a sequence of expressions that represent domain patterns for parsing your input data. Open Parser also collects statistical data and scores the parsing matches to help you determine the effectiveness of your parsing grammars.

Use Open Name Parser to:

- Parse input data using domain-specific and culture-specific parsing grammars that you define in Domain Editor.
- Parse input data using domain-independent parsing grammars that you define in Open Parser using the same simple but powerful parsing grammar available in Domain Editor.
- Parse input data using domain-independent parsing grammars at runtime that you define in Dataflow Options.
- Preview parsing grammars to test how sample input data parses before running the job using the target input data file.
- Trace parsing grammar results to view how tokens matched or did not match the expressions you defined and to better understand the matching process.

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Open Parser job.

### Table 45: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/opennameparser/input/OpenParser_Input.csv` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 46: openParserConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `OpenParser`. |
| *pb.bdq.job.name* | Name of the job. Default is `OpenParserSample`. |
| *pb.bdq.dnm.openparser.configuration* | Json string for defining open parser configuration. It specifies details, such as the parsing grammar to be used, and the language or culture of the data you want to parse. |
| *pb.bdq.reference.data* | The path where you have placed the reference data. For example, `{"referenceDataPathLocation":"LocaltoDataNodes", "dataDir":"/home/data/referenceData"}` |

### Table 47: openParserConfigHDFSRefData(DataDownloader)

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `OpenParser`. |
| *pb.bdq.job.name* | Name of the job. Default is `OpenParserSample`. |
| *pb.bdq.dnm.openparser.configuration* | Json string for defining open parser configuration. It specifies details, such as the parsing grammar to be used, and the language or culture of the data you want to parse. |
| *pb.bdq.reference.data* | Path of reference data on HDFS and the data downloader path. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/home/data/dm/referenceData", "dataDownloader":{"dataDownloader":"HDFS", "localFSRepository":"/local/download"}}` |

### Table 48: openParserConfigDistributedCache

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `OpenParser`. |
| *pb.bdq.job.name* | Name of the job. Default is `OpenParserSample`. |
| *pb.bdq.dnm.openparser.configuration* | Json string for defining open parser configuration. It specifies details, such as the parsing grammar to be used, and the language or culture of the data you want to parse. |
| *pb.bdq.reference.data* | Path of the reference data on HDFS and the type of data downloader. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/home/data/dm/referenceData", "dataDownloader":{"dataDownloader":"DC"}}` |

### Table 49: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 50: OutputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/opennameparser/output` |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |

| Parameter | Description |
|---|---|
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Table Lookup

The Table Lookup stage standardizes terms against a previously validated form of that term and applies the standard version. This evaluation is done by searching a table for the term to standardize.

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Table Lookup job.

### Table 51: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/tablelookup/input/Tablelookup_Input.txt` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |

| Parameter | Description |
|---|---|
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

**Table 52: tableLookupConfig**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `TableLookup`. |
| *pb.bdq.job.name* | Name of the job. Default is `TableLookupSample`. |
| *pb.bdq.dnm.tablelookup.configuration* | Json string for defining table lookup configuration. It specifies details, such as type of action to be taken on source field, the field containing the term you want to look up, and the table you want to use to find terms that match the data in your dataflow. |
| *pb.bdq.reference.data* | The path where you have placed the reference data. For example, `{"referenceDataPathLocation" :"LocaltoDataNodes","dataDir": "/home/data/referenceData"}` |

**Table 53: tableLookupConfigHDFSRefData(DataDownloader)**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `TableLookup`. |
| *pb.bdq.job.name* | Name of the job. Default is `TableLookupSample`. |
| *pb.bdq.dnm.tablelookup.configuration* | Json string for defining table lookup configuration. It specifies details, such as type of action to be taken on source field, the field containing the term you want to look up, and the table you want to use to find terms that match the data in your dataflow. |

| Parameter | Description |
|---|---|
| *pb.bdq.reference.data* | Path of reference data on HDFS and the data downloader path. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/home/data/dm/referenceData", "dataDownloader":{"dataDownloader":"HDFS", "localFSRepository":"/local/download"}}` |

**Table 54: tableLookupConfigDistributedCache**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `TableLookup`. |
| *pb.bdq.job.name* | Name of the job. Default is `TableLookupSample`. |
| *pb.bdq.dnm.tablelookup.configuration* | Json string for defining table lookup configuration. It specifies details, such as type of action to be taken on source field, the field containing the term you want to look up, and the table you want to use to find terms that match the data in your dataflow. |
| *pb.bdq.reference.data* | Path of the reference data on HDFS and the type of data downloader. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/home/data/dm/referenceData", "dataDownloader":{"dataDownloader":"DC"}}` |

**Table 55: mapReduceConfig**

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

**Table 56: OutputFileConfig**

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |

| Parameter | Description |
|---|---|
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/tablelookup/output`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |

| Parameter | Description |
|---|---|
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Global Addressing Module

## Global Address Validation

Global Address Validation provides enhanced address standardization and validation. Global Address Validation is part of the Global Addressing Module.

### Supported Countries

Global Address Validation provides enhanced address standardization and validation for the following prioritized countries. The three-digit ISO country code is shown for each country. For a complete list of all ISO country codes, see **ISO Country Codes and Coder Support** on page 369.

- Argentina (ARG)
- Australia (AUS)
- Austria (AUT)
- Belgium (BEL)
- Brazil (BRA)
- Canada (CAN)
- China (CHN)
- Czech Republic (CHZ)
- Denmark (DNK)
- Finland (FIN)
- France (FRA)
- Germany (DEU)
- Greece (GRC)
- India (IND)

- Ireland (IRL)
- Italy (ITA)
- Japan (JPN)
- Malaysia (MYS)
- Mexico (MEX)
- Netherlands (NLD)
- New Zealand (NZL)
- Norway (NOR)
- Poland (POL)
- Russia (RUS)
- Spain (ESP)
- Sweden (SWE)
- Switzerland (CHE)
- United Kingdom (GBR) (Includes POI information)

Global Address Validation provides additional support for 130+ countries worldwide.

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Global Address Validation job.

### Table 57: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/addressing/input/global/Global_Address.txt` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |

| Parameter | Description |
|---|---|
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

**Table 58: addressValidationConfig**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `AddressValidation`. |
| *pb.bdq.job.name* | Name of the job. Default is `AddressValidationSample`. |
| *pb.bdq.reference.data* | The path where you have placed the reference data. For example, `{"dataDir":"/home/hduser/ReferenceData/AddressQuality/GAM", "referenceDataPathLocation": "LocaltoDataNodes"}` |
| *pb.bdq.uam.addressvalidation.input.option* | Json string to define the input configurations, such as, Match Mode, Result Casing, and the Default Country. |
| *pb.bdq.uam.addressvalidation.engine.configurations* | Json string to define the engine configurations, such as, the Database Path, Country Code and the Process Type. |

**Table 59: addressValidationConfigHDFSRefData(DataDownloader)**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `AddressValidation`. |
| *pb.bdq.job.name* | Name of the job. Default is `AddressValidationSample`. |

| Parameter | Description |
|---|---|
| *pb.bdq.reference.data* | Path of reference data on HDFS and the data downloader path. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/user/root/ReferenceData/AddressValidation", "dataDownloader":{"dataDownloader":"HDFS", "localFSRepository":"/opt/PitneyBowes/ReferenceData/AddressValidation"}}` |
| *pb.bdq.uam.addressvalidation.input.option* | Json string to define the input configurations, such as, Match Mode, Result Casing, and the Default Country. |
| *pb.bdq.uam.addressvalidation.engine.configurations* | Json string to define the engine configurations, such as, the Database Path, Country Code and the Process Type. |

**Table 60: addressValidationConfigDistributedCache**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `AddressValidation`. |
| *pb.bdq.job.name* | Name of the job. Default is `AddressValidationSample`. |
| *pb.bdq.reference.data* | Path of the reference data on HDFS and the type of data downloader. For example, `{"dataDir":"/home/hduser/ReferenceData/AddressQuality/GAM", "referenceDataPathLocation":"HDFS", "dataDownloader":{"dataDownloader":"DC"}}` |
| *pb.bdq.uam.addressvalidation.input.option* | Json string to define the input configurations, such as, Match Mode, Result Casing, and the Default Country. |
| *pb.bdq.uam.addressvalidation.engine.configurations* | Json string to define the engine configurations, such as, the Database Path, Country Code and the Process Type. |

### Table 61: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 62: outputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/addressing/output/global`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| *pb.bdq.job.print.counters.console* | If the counters are printed on console or in a file. `True` indicates counters are printed on the console |
| *pb.bdq.job.counter.file.path* | Path and the name of the file to which the counters are to be printed. You need to specify this if value in the *pb.bdq.job.print.counters.console* is `false`. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |

| Parameter | Description |
|---|---|
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Universal Addressing Module

## Validate Address

Validate Address standardizes and validates addresses using postal authority address data. It can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state or province names, and more.

**Note:** Currently, Validate Address supports only US addresses.

Validate Address also returns result indicators about validation attempts, such as whether it validated the address, the level of confidence in the returned address, and the reason for failure if the address could not be validated.

During address matching and standardization, Validate Address separates address lines into components and compares those to the contents of the **Universal Addressing Module** databases. If a match is found, the input address is *standardized* to the database information. If no database match is found, Validate Address optionally *formats* the input addresses. The formatting process attempts to structure the address lines according to the conventions of the appropriate postal authority.

> **Note:** Validate Address supports CASS Certified™ processing which enables you to qualify for USPS® postal discounts.

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Validate Address job.

### Table 63: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/home/hadoop/uamus.txt` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 64: uamusConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `UniversalAddressingValidate`. |
| *pb.bdq.job.name* | Name of the job. Default is `UAMUniversalAddressingSample`. |
| *pb.bdq.reference.data* | The path where you have placed the reference data. For example, `{"dataDir":"/home/hduser/ReferenceData/AddressQuality/UAM-US", "referenceDataPathLocation":"LocaltoDataNodes"}` |
| *pb.bdq.uam.universaladdress.input.configuration* | Json string to define input configurations, such as, Process Type, Elements of Output Address, and Number of the Report Lists. |
| *pb.bdq.uam.universaladdress.general.configuration* | Json string to define general configurations, such as, the File Type, Memory Model, and Suitelink Memory Model. |
| *pb.bdq.uam.universaladdress.cobol.runtime* | The cobol runtime directory path. For example, `/home/hduser/PBSpectrum_BigDataSDK/SDK/runtime` |
| *pb.bdq.uam.universaladdress.modules.dir* | Path where the modules directory resides. For example, `/home/hduser/PBSpectrum_BigDataSDK/SDK/modules` |
| *pb.bdq.uam.universaladdress.dpv.db.path* | The path where Delivery Point Validation (DPV) database resides. For example, `/home/hduser/ReferenceData/AddressQuality/UAM/Data`<br>**Note:** This parameter is optional. |
| *pb.bdq.uam.universaladdress.ews.db.path* | The path of the Early Warning System (EWS) database. For example, `/home/hduser/ReferenceData/AddressQuality/UAM/Data`<br>**Note:** This parameter is optional. |

| Parameter | Description |
|---|---|
| *pb.bdq.uam.universaladdress.lacs.db.path* | The path where Locatable Address Conversion System (LACS) database resides. For example, `/home/hduser/ReferenceData/AddressQuality/UAM/Data`<br>**Note:** This parameter is optional. |
| *pb.bdq.uam.universaladdress.rdi.db.path* | The path where Residential Delivery Indicator (RDI) database resides. For example, `/home/hduser/ReferenceData/AddressQuality/UAM/Data`<br>**Note:** This parameter is optional. |
| *pb.bdq.uam.universaladdress.suitelink.db.path* | The suitelink database path. For example, `/home/hduser/ReferenceData/AddressQuality/UAM/Data`<br>**Note:** This parameter is optional. |
| *pb.bdq.job.report.create* | Specify `true`, if you want to generate a report on successful completion. |

**Table 65: uamusConfigHDFSRefData(DataDownloader)**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `UniversalAddressingValidate`. |
| *pb.bdq.job.name* | Name of the job. Default is `UAMUniversalAddressingSample`. |
| *pb.bdq.reference.data* | Path of reference data on HDFS and the data downloader path. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/user/root/ReferenceData/UAM-US", "dataDownloader":{"dataDownloader":"HDFS", "localFSRepository":"/opt/PitneyBowes/ReferenceData/UAM-US"}}` |

| Parameter | Description |
|---|---|
| *pb.bdq.uam.universaladdress.input.configuration* | Json string to define input configurations, such as, Process Type, Elements of Output Address, and Number of the Report Lists. |
| *pb.bdq.uam.universaladdress.general.configuration* | Json string to define general configurations, such as, the File Type, Memory Model, and Suitelink Memory Model. |
| *pb.bdq.uam.universaladdress.cobol.runtime* | The cobol runtime directory path. For example, `/home/hduser/PBSpectrum_BigDataSDK/SDK/runtime` |
| *pb.bdq.uam.universaladdress.modules.dir* | Path where the modules directory resides. For example, `/home/hduser/PBSpectrum_BigDataSDK/SDK/modules` |
| *pb.bdq.uam.universaladdress.dpv.db.path* | The path where Delivery Point Validation (DPV) database resides. For example, `hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/Data.zip`<br>**Note:** This parameter is optional. |
| *pb.bdq.uam.universaladdress.ews.db.path* | The path of the Early Warning System (EWS) database. For example, `hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/Data.zip`<br>**Note:** This parameter is optional. |
| *pb.bdq.uam.universaladdress.lacs.db.path* | The path where Locatable Address Conversion System (LACS) database resides. For example, `hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/Data.zip`<br>**Note:** This parameter is optional. |
| *pb.bdq.uam.universaladdress.rdi.db.path* | The path where Residential Delivery Indicator (RDI) database resides. For example, `hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/RDI.zip`<br>**Note:** This parameter is optional. |

| Parameter | Description |
|---|---|
| *pb.bdq.uam.universaladdress.suitelink.db.path* | The suitelink database path. For example, `hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/Data.zip`<br>**Note:** This parameter is optional. |
| *pb.bdq.job.report.create* | Specify `true`, if you want to generate a report on successful completion. |

**Table 66: uamusConfigDistributedCache**

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `UniversalAddressingValidate`. |
| *pb.bdq.job.name* | Name of the job. Default is `UAMUniversalAddressingSample`. |
| *pb.bdq.reference.data* | Path of the reference data on HDFS and the type of data downloader. For example, `{"dataDir":"/user/hduser/ReferenceData/AddressQuality/UAM", "referenceDataPathLocation":"HDFS", "dataDownloader":{"dataDownloader":"DC"}}` |
| *pb.bdq.uam.universaladdress.input.configuration* | Json string to define input configurations, such as, Process Type, Elements of Output Address, and Number of the Report Lists. |
| *pb.bdq.uam.universaladdress.general.configuration* | Json string to define general configurations, such as, the File Type, Memory Model, and Suitelink Memory Model. |
| *pb.bdq.uam.universaladdress.acushare.license* | The path where you have placed the acushare license file. For example, `/home/hduser/runcbl.alc` |
| *pb.bdq.uam.universaladdress.acushare.service* | A `true` value indicates that acushare service is running. |
| *pb.bdq.uam.universaladdress.unix.version* | Specifies the Unix version of your cluster node. For example, `REDHAT7`. |

| Parameter | Description |
|---|---|
| *pb.bdq.uam.universaladdress.cobol.runtime* | The cobol runtime directory path. For example, `/home/hduser/PBSpectrum_BigDataSDK/SDK/runtime` |
| *pb.bdq.uam.universaladdress.modules.dir* | Path where the modules directory resides. For example, `/home/hduser/PBSpectrum_BigDataSDK/SDK/modules` |
| *pb.bdq.uam.universaladdress.dpv.db.path* | The path where Delivery Point Validation (DPV) database resides. For example, `/home/hduser/ReferenceData/AddressQuality/UAM/Data`<br>**Note:** This parameter is optional. |
| *pb.bdq.uam.universaladdress.ews.db.path* | The path of the Early Warning System (EWS) database. For example, `/home/hduser/ReferenceData/AddressQuality/UAM/Data`<br>**Note:** This parameter is optional. |
| *pb.bdq.uam.universaladdress.lacs.db.path* | The path where Locatable Address Conversion System (LACS) database resides. For example, `/home/hduser/ReferenceData/AddressQuality/UAM/Data`<br>**Note:** This parameter is optional. |
| *pb.bdq.uam.universaladdress.rdi.db.path* | The path where Residential Delivery Indicator (RDI) database resides. For example, `/home/hduser/ReferenceData/AddressQuality/UAM/Data`<br>**Note:** This parameter is optional. |
| *pb.bdq.uam.universaladdress.suitelink.db.path* | The suitelink database path. For example, `/home/hduser/ReferenceData/AddressQuality/UAM/Data`<br>**Note:** This parameter is optional. |

| Parameter | Description |
|---|---|
| *pb.bdq.job.report.create* | Specify `true`, if you want to generate a report on successful completion. |

### Table 67: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 68: outputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/home/hadoop/output`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| *pb.bdq.job.print.counters.console* | If the counters are printed on console or in a file. `True` indicates counters are printed on the console |
| *pb.bdq.job.counter.file.path* | Path and the name of the file to which the counters are to be printed. You need to specify this if value in the *pb.bdq.job.print.counters.console* is `false`. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |

| Parameter | Description |
|---|---|
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Validate Address Global

Validate Address Global provides enhanced address standardization and validation for addresses outside the U.S. and Canada. Validate Address Global can also validate addresses in the U.S. and Canada but its strength is validation of addresses in other countries. If you process a significant number of addresses outside the U.S. and Canada, you should consider using Validate Address Global.

Validate Address Global is part of the Universal Addressing Module.

Validate Address Global performs several steps to achieve a quality address, including parsing, validation, and formatting.

### *Address Parsing, Formatting, and Standardization*

Restructuring incorrectly fielded address data is a complex and difficult task especially when done for international addresses. People introduce many ambiguities as they enter address data into computer systems. Among the problems are misplaced elements (such as company or personal names in street address fields) or varying abbreviations that are not only language, but also country specific. Validate Address Global identifies address elements in address lines and assigns them to the proper fields. This is an important precursor to the actual validation. Without restructuring, "no match" situations might result.

Properly identified address elements are also important when addresses have to be truncated or shortened to fit specific field length requirements. With the proper information in the right fields, specific truncation rules can be applied.

- Parses and analyzes address lines and identifies individual address elements
- Processes over 30 different character sets
- Formats addresses according to the postal rules of the country of destination
- Standardizes address elements (such as changing AVENUE to AVE)

### *Global Address Validation*

Address validation is the correction process where properly parsed address data is compared against reference databases supplied by postal organizations or other data providers. Validate Address Global validates individual address elements to check for correctness using sophisticated fuzzy matching technology and produces standardized and formatted output based on postal standards and user preferences. FastCompletion validation type can be used in quick address entry applications. It allows input of truncated data in several address fields and generates suggestions based on this input.

In some cases, it is not possible to fully validate an address. Here Validate Address Global has a unique deliverability assessment feature that classifies addresses according to their probable deliverability.

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Validate Address Global job.

**Table 69: inputFileConfig**

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |

| Parameter | Description |
|---|---|
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/addressing/input/global/Global_Address.txt` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 70: globalAddressingConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `GlobalAddressingValidate`. |
| *pb.bdq.job.name* | Name of the job. Default is `GlobalAddressingValidateSample`. |
| *pb.bdq.reference.data* | The path where you have placed the reference data. For example, `{"dataDir":"/home/hduser/ReferenceData/AddressQuality/Global", "referenceDataPathLocation":"LocaltoDataNodes"}` |
| *pb.bdq.uam.global.engine.configurations.preload* | Preload type in the global engine configuration. The values can be: `NONE`, `FULL`, or `PARTIAL`. |
| *pb.bdq.uam.global.engine.configurations.database.type* | Database Type in the global engine configuration. The values can be `BATCH_INTERACTIVE`, `FASTCOMPLETION`, or `CERTIFIED`. |

| Parameter | Description |
|---|---|
| *pb.bdq.uam.global.engine.configurations.supported.countries* | Supported countries for global address validation job, such as United States Of America, Great Britain, and Canada.<br><br>**Note:** You can specify multiple countries as comma-separated values. |
| *pb.bdq.uam.global.input.configuration* | Json string to define the input configurations, such as, Match Mode, Default Country, Maximum Results, Result Casing, State Province Type, and Optimization Level. |
| *pb.bdq.uam.global.general.configuration* | Json string to define general configuration, such as Cache Size, Maximum Thread Count, and Maximum Limit of Memory Usage. |
| *pb.bdq.uam.global.unlockCode* | Code to unlock data in the database. |

### Table 71: globalAddressingConfigHDFSRefData(DataDownloader)

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `GlobalAddressingValidate`. |
| *pb.bdq.job.name* | Name of the job. Default is `GlobalAddressingValidateSample`. |
| *pb.bdq.reference.data* | Path of reference data on HDFS and the data downloader path. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/user/root/ReferenceData/Global/Global.zip", "dataDownloader":{"dataDownloader":"HDFS", "localFSRepository":"/opt/PitneyBowes/ReferenceData/GlobalAddress"}}` |
| *pb.bdq.uam.input.groupby.region* | Specifies if the input address data should be grouped by region, such as *APAC*, *EMEA*, and *America*.<br><br>A `true` value indicates grouping.<br><br>**Note:** This parameter is applicable only if you have placed your reference data on HDFS. |

| Parameter | Description |
|---|---|
| *pb.bdq.uam.global.engine.configurations.preload* | Preload type in the global engine configuration. The values can be: `NONE`, `FULL`, or `PARTIAL`. |
| *pb.bdq.uam.global.engine.configurations.database.type* | Database Type in the global engine configuration. The values can be `BATCH_INTERACTIVE`, `FASTCOMPLETION`, or `CERTIFIED`. |
| *pb.bdq.uam.global.engine.configurations.supported.countries* | Supported countries for global address validation job, such as United States Of America, Great Britain, and Canada.<br>**Note:** You can specify multiple countries as comma-separated values. |
| *pb.bdq.uam.global.input.configuration* | Json string to define the input configurations, such as, Match Mode, Default Country, Maximum Results, Result Casing, State Province Type, and Optimization Level. |
| *pb.bdq.uam.global.general.configuration* | Json string to define general configuration, such as Cache Size, Maximum Thread Count, and Maximum Limit of Memory Usage. |
| *pb.bdq.uam.global.unlockCode* | Code to unlock data in the database. |

### Table 72: globalAddressingConfigDistributedCache

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `GlobalAddressingValidate`. |
| *pb.bdq.job.name* | Name of the job. Default is `GlobalAddressingValidateSample`. |
| *pb.bdq.reference.data* | Path of the reference data on HDFS and the type of data downloader. For example, `{"dataDir":"/home/hduser/ReferenceData/AddressQuality/Global", "referenceDataPathLocation":"HDFS", "dataDownloader":{"dataDownloader":"DC"}}` |
| *pb.bdq.uam.global.engine.configurations.preload* | Preload type in the global engine configuration. The values can be: `NONE`, `FULL`, or `PARTIAL`. |

| Parameter | Description |
|---|---|
| *pb.bdq.uam.global.engine.configurations.database.type* | Database Type in the global engine configuration. The values can be `BATCH_INTERACTIVE`, `FASTCOMPLETION`, or `CERTIFIED`. |
| *pb.bdq.uam.global.engine.configurations.supported.countries* | Supported countries for global address validation job, such as United States Of America, Great Britain, and Canada.<br>**Note:** You can specify multiple countries as comma-separated values. |
| *pb.bdq.uam.global.input.configuration* | Json string to define the input configurations, such as, Match Mode, Default Country, Maximum Results, Result Casing, State Province Type, and Optimization Level. |
| *pb.bdq.uam.global.general.configuration* | Json string to define general configuration, such as Cache Size, Maximum Thread Count, and Maximum Limit of Memory Usage. |
| *pb.bdq.uam.global.unlockCode* | Code to unlock data in the database. |

### Table 73: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 74: outputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/addressing/output/global`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |

| Parameter | Description |
|---|---|
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| *pb.bdq.job.print.counters.console* | If the counters are printed on console or in a file. `True` indicates counters are printed on the console |
| *pb.bdq.job.counter.file.path* | Path and the name of the file to which the counters are to be printed. You need to specify this if value in the *pb.bdq.job.print.counters.console* is `false`. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |

| Parameter | Description |
|---|---|
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Validate Address Loqate

Validate Address Loqate standardizes and validates addresses using postal authority address data. Validate Address Loqate can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state/province names.

Validate Address Loqate also returns result indicators about validation attempts, such as whether or not Validate Address Loqate validated the address, the level of confidence in the returned address, the reason for failure if the address could not be validated, and more.

During address matching and standardization, Validate Address Loqate separates address lines into components and compares them to the contents of the Universal Addressing Module databases. If a match is found, the input address is standardized to the database information. If no database match is found, ValidateAddress Loqate optionally formats the input addresses. The formatting process attempts to structure the address lines according to the conventions of the appropriate postal authority.Validate Address Loqate is part of the Universal Addressing Module.

## Configuration Files

These tables describe the parameters and the values you need to specify before you run the Validate Address Loqate job.

### Table 75: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/addressing/input/loqate/loqate_input.txt` |

| Parameter | Description |
|---|---|
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 76: loqateAddressingConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `LoqateAddressingValidate`. |
| *pb.bdq.job.name* | Name of the job. Default is `LoqateAddressingValidateSample`. |
| *pb.bdq.reference.data* | The path where you have placed the reference data. For example, `{"dataDir":"/home/hduser/ReferenceData/AddressQuality/Loqate", "referenceDataPathLocation":"LocaltoDataNodes"}` |
| *pb.bdq.uam.loqate.process.configuration* | Json string to define validate configurations, such as, the Process Type, Minimum Match Score, Default Country and the Level Of Acceptance. |
| *pb.bdq.uam.loqate.engine.configuration* | Json string to define engine configurations, such as, Tool Info, log File Name, and Match Score Threshold Factor. |
| *pb.bdq.uam.loqate.general.configuration* | Json string to define general configurations, such as, Maximum Idle Objects, Minimum Idle Objects, and Maximum Wait Time. |

### Table 77: loqateAddressingConfigHDFSRefData(DataDownloader)

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `LoqateAddressingValidate`. |
| *pb.bdq.job.name* | Name of the job. Default is `LoqateAddressingValidateSample`. |
| *pb.bdq.reference.data* | Path of reference data on HDFS and the data downloader path. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/user/root/ReferenceData/Loqate", "dataDownloader":{"dataDownloader":"HDFS", "localFSRepository":"/opt/PitneyBowes/ReferenceData/Loqate"}}` |
| *pb.bdq.uam.loqate.process.configuration* | Json string to define validate configurations, such as, the Process Type, Minimum Match Score, Default Country and the Level Of Acceptance. |
| *pb.bdq.uam.loqate.engine.configuration* | Json string to define engine configurations, such as, Tool Info, log File Name, and Match Score Threshold Factor. |
| *pb.bdq.uam.loqate.general.configuration* | Json string to define general configurations, such as, Maximum Idle Objects, Minimum Idle Objects, and Maximum Wait Time. |

### Table 78: loqateAddressingConfigDistributedCache

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `LoqateAddressingValidate`. |
| *pb.bdq.job.name* | Name of the job. Default is `LoqateAddressingValidateSample`. |

| Parameter | Description |
|---|---|
| *pb.bdq.reference.data* | Path of the reference data on HDFS and the type of data downloader. For example, `{"dataDir":"/home/hduser/ReferenceData/AddressQuality/Loqate", "referenceDataPathLocation":"HDFS", "dataDownloader":{"dataDownloader":"DC"}}` |
| *pb.bdq.uam.loqate.process.configuration* | Json string to define validate configurations, such as, the Process Type, Minimum Match Score, Default Country and the Level Of Acceptance. |
| *pb.bdq.uam.loqate.engine.configuration* | Json string to define engine configurations, such as, Tool Info, log File Name, and Match Score Threshold Factor. |
| *pb.bdq.uam.loqate.general.configuration* | Json string to define general configurations, such as, Maximum Idle Objects, Minimum Idle Objects, and Maximum Wait Time. |

### Table 79: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 80: outputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/addressing/output/loqate`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |

| Parameter | Description |
|---|---|
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| *pb.bdq.job.print.counters.console* | If the counters are printed on console or in a file. `True` indicates counters are printed on the console |
| *pb.bdq.job.counter.file.path* | Path and the name of the file to which the counters are to be printed. You need to specify this if value in the *pb.bdq.job.print.counters.console* is `false`. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |

| Parameter | Description |
|---|---|
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# Universal Name Module

## OpenNameParser

OpenNameParser breaks down personal and business names and other terms in the name data field into their component parts. These parsed name elements are then subsequently available to other automated operations such as name matching, name standardization, or multi-record name consolidation.

OpenNameParser does the following:

- Determines the type of a name in order to describe the function that the name performs. Name entity types are divided into two major groups: personal names and business names. Within each of these major groups are subgroups.
- Determines the form of a name in order to understand which syntax the parser should follow for parsing. Personal names usually take on a natural (signature) order or a reverse order. Business names are usually ordered hierarchically.
- Determines and labels the component parts of a name so that the syntactical relationship of each name part to the entire name is identified. The personal name syntax includes prefixes, first, middle, and last name parts, suffixes, and account description terms, among other personal name parts. The business name syntax includes the firm name and suffix terms.
- Parses conjoined personal and business names and either retains them as one record or splits them into multiple records. Examples of conjoined names include "Mr. and Mrs. John Smith" and "Baltimore Gas & Electric dba Constellation Energy".
- Parses output as records or as a list.
- Assigns a parsing score that reflects the degree of confidence that the parsing is correct.

### Configuration Files

These tables describe the parameters and the values you need to specify before you run the Open Name Parser job.

### Table 81: inputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.input.type* | Input file type. The values can be: `TEXT`, `ORC` or `PARQUET`. |
| *pb.bdq.inputfile.path* | The path where you have placed the input file on HDFS. For example, `/user/hduser/sampledata/opennameparser/input/OpenNameParser_Input.csv` |
| *textinputformat.record.delimiter* | File record delimiter used in the text type input file. For example, `LINUX`, `MACINTOSH`, or `WINDOWS` |
| *pb.bdq.inputformat.field.delimiter* | Field or column delimiter used in the input file, such as comma (`,`) or tab. |
| *pb.bdq.inputformat.text.qualifier* | Text qualifiers, if any, in the columns or fields of the input file. |
| *pb.bdq.inputformat.file.header* | Comma-separated value of the headers used in the input file. |
| *pb.bdq.inputformat.skip.firstrow* | If the first row is to be skipped from processing. The values can be `True` or `False`, where `True` indicates skip. |

### Table 82: openNameParserConfig

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `OpenNameParser`. |
| *pb.bdq.job.name* | Name of the job. Default is `OpenNameParserSample`. |
| *pb.dq.unm.opennameparser.configuration* | Json string to define open name parser configurations, such as, types of names to be parsed. |
| *pb.bdq.reference.data* | The path where you have placed the reference data. For example, `{"referenceDataPathLocation":"LocaltoDataNodes", "dataDir":"/home/data/referenceData"}` |

### Table 83: openNameParserConfigHDFSRefData(DataDownloader)

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `OpenNameParser`. |
| *pb.bdq.job.name* | Name of the job. Default is `OpenNameParserSample`. |
| *pb.dq.unm.opennameparser.configuration* | Json string to define open name parser configurations, such as, types of names to be parsed. |
| *pb.bdq.reference.data* | Path of reference data on HDFS and the data downloader path. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/home/data/dm/referenceData", "dataDownloader":{"dataDownloader":"HDFS", "localFSRepository":"/local/download"}}` |

### Table 84: openNameParserConfigDistributedCache

| Parameter | Description |
|---|---|
| *pb.bdq.job.type* | This is a constant value that defines the job. The value for this job is: `OpenNameParser`. |
| *pb.bdq.job.name* | Name of the job. Default is `OpenNameParserSample`. |
| *pb.dq.unm.opennameparser.configuration* | Json string to define open name parser configurations, such as, types of names to be parsed. |
| *pb.bdq.reference.data* | Path of the reference data on HDFS and the type of data downloader. For example, `{"referenceDataPathLocation":"HDFS", "dataDir":"/home/data/dm/referenceData", "dataDownloader":{"dataDownloader":"DC"}}` |

### Table 85: mapReduceConfig

| Specifies the MapReduce configuration parameters |
|---|
| Use this file to customize MapReduce parameters, such as *mapreduce.map.memory.mb*, *mapreduce.reduce.memory.mb* and *mapreduce.map.speculative*, as needed for your job. |

### Table 86: outputFileConfig

| Parameter | Description |
|---|---|
| *pb.bdq.output.type* | Specify if the output is in: `TEXT`, `ORC`, or `PARQUET` format. |
| *pb.bdq.outputfile.path* | The path where you want the output file to be generated on HDFS. For example, `/user/hduser/sampledata/opennameparser/output`. |
| *pb.bdq.outputformat.field.delimiter* | Field or column delimiter in the output file, such as comma (`,`) or tab. |
| *pb.bdq.output.overwrite* | For a `true` value, the output folder is overwritten every time job is run. |
| *pb.bdq.outputformat.headerfile.create* | Specify `true`, if the output file needs to have a header. |
| *pb.bdq.job.print.counters.console* | If the counters are printed on console or in a file. `True` indicates counters are printed on the console |
| *pb.bdq.job.counter.file.path* | Path and the name of the file to which the counters are to be printed. You need to specify this if value in the *pb.bdq.job.print.counters.console* is `false`. |
| **Properties of Parquet file** | |
| *parquet.compression* | The compression algorithm used to compress pages. It is one of these: `UNCOMPRESSED`, `SNAPPY`, `GZIP`, or `LZO`.<br>Default is `UNCOMPRESSED`. |
| *parquet.block.size* | The size of a row group being buffered in memory.<br>Larger values improve the *I/O* when reading but consume more memory when writing.<br>Default size is 134217728 bytes (= 128 * 1024 * 1024) |
| *parquet.page.size* | Page constitutes block and is the smallest unit that must be read fully to access a single record.<br>Default size is 1048576 bytes (= 1 * 1024 * 1024)<br>**Note:** A very small page size results in deterioration of compression. |

| Parameter | Description |
|---|---|
| *parquet.dictionary.page.size* | Default size is 1048576 bytes (= 1 * 1024 * 1024) |
| *parquet.enable.dictionary* | The boolean value (`True` or `False`) to enable or disable dictionary encoding. Default is `True` |
| *parquet.validation* | Default boolean value is `False`. |
| *parquet.writer.version* | Specifies the version of writer. It should be `PARQUET_1_0` or `PARQUET_2_0`. Default is `PARQUET_1_0`. |
| *parquet.writer.max-padding* | Default to no padding, 0% of the row group size |
| *parquet.page.size.check.estimate* | Default boolean value is `True` |
| *parquet.page.size.row.check.min* | Default is 100 |
| *parquet.page.size.row.check.max* | Default is 10000 |

# 6 - Hive User-Defined Functions

## In this section

# Introduction

Apache Hive provides User Defined Functions (UDF). A UDF can be defined to perform required actions and achieve desired objectives.

The Spectrum™ Data & Address Quality for Big Data SDK provides a set of Hive User Defined Functions and User Defined Aggregation Functions to run the listed Data Quality jobs.

### *User Defined Functions (UDF)*

A User Defined Function processes one record at a time.

The UDF based jobs are:

- Advanced Transformer
- Custom Groovy Script
- Global Address Validation
- Match Key Generator
- Open Name Parser
- Open Parser
- Table Lookup
- Validate Address
- Validate Address Global
- Validate Address Loqate

### *User Defined Aggregation Functions (UDAF)*

A User Defined Aggregation Function first aggregates records into collections based on the join field, and then processes one collection of records at a time.

The UDAF based jobs are:

- Best of Breed
- Duplicate Synchronization
- Filter
- Interflow Match
- Intraflow Match
- Transactional Match

## Components of a Hive Function

The key components required to run a Spectrum™ Data & Address Quality for Big Data SDK Hive UDF are:

| | |
|---|---|
| **JAR File** | The Spectrum™ Data & Address Quality for Big Data SDK Hive JAR file of the module to which the desired Data Quality Hive UDF belongs. This must be registered before using any UDF. |
| **Job UDF / UDAF** | Each Data Quality job is provided as either a User Defined Function (UDF) or a User Defined Aggregation Function (UDAF). |
| **Alias** | The alias assigned to a Hive UDF. This is optional. |
| **Configurations** | The rules specified in JSON format, and other configuration details, based on which the job is to be run. |
| **Reference Data** | The reference data can reside on Hadoop Distributed File System (HDFS) or locally on cluster machines.<br>On HDFS, the reference data can be in any of these two formats:<br>• As files<br>• As archive<br>In case of local placement, the reference data must be present on each node of the cluster at the same path. |
| **Header** | The header fields of the input table, in comma-separated format. |
| **Input Table** | The table which provides the input records respectively for the Hive UDF to be run. |
| **Candidate Table** | The table which provides the candidate records for the Hive UDF to be run, in case of the Interflow Match UDAF. |
| **Suspect Table** | The table which provides the suspect records for the Hive UDF to be run, in case of Interflow Match UDAF. |
| **hive.fetch.task.conversion** | To convert select queries to a single FETCH task, minimizing latency.<br>Set the value to `none` or `minimal`. Default is `minimal`.<br>**Note:** This configuration is required for all UDFs. |
| **hive.map.aggr** | To turn the aggregation of data between Mapper and Reducer on or off, set this Hive environment variable to `false`. By default, it is `true` and the data is aggregated.<br>Set this value to `false` for all Hive jobs in the SDK.<br>**Note:** This configuration is required for all UDAFs. |
| **General Configurations** | The memory configurations required to run the job.<br>**Note:** This configuration is required only for Universal Addressing Module Hive UDAFs. |

**Input Configurations** The settings for the input data.

> **Note:** This configuration is required only for Universal Addressing Module Hive UDAFs.

**Engine Configurations** To set various configurations, such as database settings, *COBOL runtime path*, *preloading type*.

> **Note:** This configuration is required only for Universal Addressing Module Hive UDAFs.

**LD_LIBRARY_PATH** To set this environment variable to the paths of the various COBOL libraries required while running the Hive jobs.

> **Note:** This configuration is required only for the Validate Address Hive UDF.

**Process Type** To specify the desired validation level to be used in a particular Hive job of the SDK. Currently, only address validation is supported.

Set this value to `VALIDATE`.

> **Note:** This configuration is required only for the Validate Address and Validate Address Loqate Hive UDAFs.

**Output** The output of the Hive UDF, which may be displayed on the console or dumped to an output file.

**Query** The query to run the required Hive UDF.

For each job, you can achieve any of these using the applicable query syntax:

- Display the output of the job on the console.
- Dump the output of the job in a designated output file.

## Using a Hive UDF

To run each Hive UDF job, you can either run these steps individually on your Hive client within a single session, or create an HQL file compiling all the required steps sequentially and run it in one go.

1. In your Hive client, log in to the required Hive database.
2. Register the JAR file of the particular Spectrum™ Data & Address Quality for Big Data SDK Module to which the desired Data Quality Hive UDF belongs.

3. In case of the Validate Address UDF, to set the path of the COBOL libraries, set the environment variable `LD_LIBRARY_PATH` as:

```
set mapreduce.admin.user.env =
LD_LIBRARY_PATH=/home/hduser/~/runtime/lib:
/home/hduser/~/runtime/bin:/home/hduser/~/server/modules/universaladdress/lib,
 ACU_RUNCBL_JNI_ONLOAD_DISABLE=1, G1RTS=/home/hduser/~/ ;
```

4. In case of the Validate Address Global UDF, add the file *libAddressDoctor5.so* file as well.
5. In case of the Validate Address Loqate UDF, add these required files to the distributed cache.
   - `loqate-core.car`
   - `LoqateVerificationLevel.csv`
   - `Loqate.csv`
   - `countryTables.csv`
   - `countryNameTables.csv`
6. Create an alias for the Hive UDF of the Data Quality job you wish to run.
   For example:

```
CREATE TEMPORARY FUNCTION matchkeygenerator as
'com.pb.bdq.amm.process.hive.matchkeygenerator.MatchKeyGeneratorUDF';
```

7. Specify the reference data path.
   - If the reference data is on HDFS, add the reference data and set reference directory as shown in this example.

     If the reference data is in *file* format:

```
hdfs://<HOST>:<PORT>/home/hduser/Refdata/;
set hivevar:refdir='./Refdata';
```

     If the reference data is in *archive* format:

```
hdfs://<HOST>:<PORT>/home/hduser/ref.zip;
set hivevar:refdir='./ref.zip';
```

   - If the reference data is on local path, ensure data is present on each node of the cluster on the same path.

     Set the reference directory as shown:

```
set hivevar:refdir='/home/hadoop/reference/';
```

8. Specify the configurations, such as the match rule, sort field, express match column, and other details for the job and assign to respective variable or configuration properties.

**Note:** The rule must be in JSON format.

For example

```
set rule='{"matchKeys":[{"expressMatchKey":false,
"matchKeyField":"MatchKey1",
"rules":[{"algorithm":"Soundex"," field":"businessname",
"startPosition":1, "length":0,"active":true, "sortInput":null,
 "removeNoiseCharacters":false}]},
{"expressMatchKey":false, "matchKeyField":"MatchKey2",
"rules":[{"algorithm":"Koeln", "field":"businessname",
"startPosition":1, "length":0, "active":true, "sortInput":null,
 "removeNoiseCharacters":false}]}]}';
```

**Note:** Use the configuration properties in the respective job configurations. For example, `pb.bdq.match.rule`, `pb.bdq.match.express.column`, and `pb.bdq.consolidation.sort.field`, wherever indicated in the respective sample HQL files.

9. Specify the header fields of the input table, in comma-separated format, and assign to a variable or configuration property.

```
set pb.bdq.match.header='businessname,recordid';
```

**Note:** Use the configuration property where indicated in the HQL file. For example, `pb.bdq.match.header` and `pb.bdq.consolidation.header`.

10. Switch off the aggregatiion of data between Reducer and Mapper, by seting the `Hive.Map.Aggr` environment variable configuration to `false`, as indicated in this example:

```
set hive.map.aggr = false;
```

**Note:** This configuration is required for all UDFs.

11. Set the general configurations for running the job as indicated in this example:

```
set pb.bdq.uam.universaladdress.general.configuration =
{"dFileType":"SPLIT", "dMemoryModel":"MEDIUM",
"lacsLinkMemoryModel":"MEDIUM", "suiteLinkMemoryModel":"MEDIUM"};
```

**Note:** This configuration is required only for Universal Addressing Module Hive UDAFs.

12. Set the input configurations for running the job as indicated in this example:

```
set pb.bdq.uam.universaladdress.input.configuration =
{"outputStandardAddress":true, "outputPostalData":false,
"outputParsedInput":false,
```

```
 "outputAddressBlocks":true, "performUSProcessing":true,
"performCanadianProcessing":
false, "performInternationalProcessing":false,
"outputFormattedOnFail":false,
"outputCasing":"MIXED", "outputPostalCodeSeparator":true,
"outputMultinationalCharacters":
false, "performDPV":false, "performRDI":false, "performESM":false,
"performASM":false,
"performEWS":false, "performLACSLink":false, "performLOT":false,
"failOnCMRAMatch":false,
 "extractFirm":false, "extractUrb":false, "outputReport3553":false,
"outputReportSERP":false,
 "outputReportSummary":true, "outputCASSDetail":false,
"outputFieldLevelReturnCodes":false,
 "keepMultimatch":false, "maximumResults":10, "standardAddressFormat":
"STANDARD_ADDRESS_FORMAT_COMBINED_UNIT", "standardAddressPMBLine":
"STANDARD_ADDRESS_PMB_LINE_NONE",
"cityNameFormat":"CITY_FORMAT_STANDARD",
 "vanityCityFormatLong":true, "outputCountryFormat":"ENGLISH",
"homeCountry":
"United States",
"streetMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"firmMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"directionalMatchingStrictness":
"MATCHING_STRICTNESS_MEDIUM", "dualAddressLogic":"DUAL_NORMAL",
"dpvSuccessfulStatusCondition":"A", "reportListFileName":"",
"reportlistProcessorName":"",
 "reportlistNumber":1, "reportMailerAddress":"", "reportMailerName":"",

"reportMailerCityLine":"", "canReportMailerCPCNumber":"",
"canReportMailerAddress":"",
 "canReportMailerName":"", "canReportMailerCityLine":"",
"internationalCityStreetSearching"
:100, "addressLineSearchOnFail":true, "outputStreetAlias":true,
"outputVeriMoveBlock":false,
 "dpvDetermineNoStat":false, "dpvDetermineVacancy":false,
"outputAbbreviatedAlias":false,
 "outputPreferredAlias":false,
"outputPreferredCity":"CITY_OVERRIDE_NAME_ZIP4",
"performSuiteLink":false, "suppressZplusPhantomCarrierR777":false,
"canStandardAddressFormat"
:"D", "canEnglishApartmentLabel":"APT",
"canFrenchApartmentLabel":"APP", "canFrenchFormat":
"C", "canOutputCityFormat":"D", "canOutputCityAlias":true,
"canDualAddressLogic":"D",
"canPreferHouseNum":false, "canSSLVRFLG":false,
"canRuralRouteFormat":"A",
"canNonCivicFormat":"A", "canDeliveryOfficeFormat":"I",
"canEnableSERP":false,
"canSwitchManagedPostalCodeConfidence":false, "stats":null,
"counts":null, "z3seg":null,
 "serpStats":null, "dpvSeedList":null, "lacsSeedList":null,
"zipInputSet":null, "reportName"
```

```
:null, "currentUser":null, "jobName":null, "jobId":null,
"jobRequest":false, "properties":
{"DPVDetermineVacancy":"N", "DualAddressLogic":"N", "ExtractUrb":"N",
 "CanFrenchFormat"
:"C", "AddressLineSearchOnFail":"Y", "OutputFieldLevelReturnCodes":"N",

"OutputFormattedOnFail":"N", "OutputStreetNameAlias":"Y",
"OutputReportSERP":"N",
 "OutputAddressBlocks":"Y", "ExtractFirm":"N",
"CanEnglishApartmentLabel":"APT",
"OutputPreferredCity":"Z", "FirmMatchingStrictness":"M",
"CanFrenchApartmentLabel":"APP",
 "KeepMultimatch":"N", "StandardAddressPMBLine":"N",
"PerformSuiteLink":"N",
"CanStandardAddressFormat":"D", "DPVSuccessfulStatusCondition":"A",
"PerformLACSLink":"N",
 "PerformUSProcessing":"Y", "PerformEWS":"N",
"StandardAddressFormat":"C",
"SuppressZplusPhantomCarrierR777":"N", "HomeCountry":"United States",

"ReportMailerAddress":"", "OutputReport3553":"N",
"OutputVeriMoveDataBlock":"N",
 "CanDeliveryOfficeFormat":"I", "OutputAbbreviatedAlias":"N",
"PerformCanadianProcessing":
"N", "PerformDPV":"N", "PerformInternationalProcessing":"N",
"CanSSLVRFlg":"N",
"StreetMatchingStrictness":"M",
"InternationalCityStreetSearching":"100",
"canSwitchManagedPostalCodeConfidence":"N", "CanDualAddressLogic":"D",
 "PerformASM":
"N", "OutputCasing":"M", "ReportListFileName":"",
"CanReportMailerAddress":"",
"ReportMailerCityLine":"", "CanReportMailerCPCNumber":"",
"ReportListProcessorName":"",
 "CanOutputCityAlias":"Y", "DirectionalMatchingStrictness":"M",
"CanRuralRouteFormat":
"A", "CanOutputCityFormat":"D", "ReportListNumber":"1",
"CanReportMailerCityLine":"",
 "OutputMultinationalCharacters":"N", "EnableSERP":"N",
"CanNonCivicFormat":"A",
"OutputShortCityName":"S", "OutputPostalCodeSeparator":"Y",
"FailOnCMRAMatch":"N",
"PerformLOT":"N", "OutputCountryFormat":"E", "CanPreferHouseNum":"N",

"CanReportMailerName":"", "PerformRDI":"N", "ReportMailerName":"",
"PerformESM":"N",
 "OutputReportSummary":"Y", "OutputVanityCityFormatLong":"Y",
"OutputPreferredAlias":"N",
 "DPVDetermineNoStat":"N", "MaximumResults":"10"}}};
```

**Note:** This configuration is required only for Universal Addressing Module Hive UDAFs.

13. Set the engine configurations for running the job as indicated in this example:

```
set pb.bdq.uam.universaladdress.engine.configurations = {
"referenceData":{
"dataDir":"/home/hduser/resources/uam/universaladdress/UAM_universaladdress4.0_Feb15/",
 "referenceDataPathLocation":"LocaltoDataNodes"},
"cobolRuntimePath":"/home/hduser/addressquality/",
"modulesDir":"/home/hduser/tapan/addressquality/modules",
"dpvDbPath":null, "suiteLinkDBPath":null, "ewsDBPath":null,
"rdiDBPath":null, "lacsDBPath":null};
```

**Note:** This configuration is required only for Universal Addressing Module Hive UDAFs.

14. Set the process type to indicate the desired validation level. We currently support address validation only.

    For example, in the *Validate Address* job, set the *process type* as below:

```
set pb.bdq.uam.universaladdress.process.type=VALIDATE;
```

**Note:** This configuration is required only for the Validate Address and Validate Address Loqate Hive UDAFs.

15. To run the job and display the job output on the console, write the query as indicated in this example:

```
SELECT businessname, recordid, bar.ret["MatchKey1"] AS MatchKey1,
bar.ret["MatchKey2"] AS MatchKey2 FROM (
SELECT *, matchkeygenerator (${hiveconf:rule}, ${hiveconf:header},
businessname, recordid) AS ret FROM cust ) bar;
```

    To run the job and dump the job output in a designated file, write the query as indicated in the below example:

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/MatchKey/' row format
 delimited FIELDS TERMINATED BY ',' MAP FIELDS TERMINATED BY ':'
COLLECTION ITEMS TERMINATED BY '|' LINES TERMINATED BY '\n' STORED AS
 TEXTFILE
SELECT businessname, recordid, bar.ret["MatchKey1"] AS MatchKey1,
bar.ret["MatchKey2"] AS MatchKey2 FROM (
SELECT *, matchkeygenerator (${hiveconf:rule}, ${hiveconf:header},
businessname, recordid) AS ret FROM cust ) bar;
```

**Note:** Ensure to use the alias defined earlier for the UDF.

**Important:** For all UDAF jobs, use the respective configuration properties as variables while defining the input parameters, where indicated in the respective sample HQL files.

For example, `pb.bdq.match.rule`, `pb.bdq.match.express.column` and `pb.bdq.consolidation.sort.field`.

# Advanced Matching Module Functions

## Using a Hive UDF of Advance Matching Module

To run each Hive UDF job, you can either run these steps individually on your Hive client within a single session, or create an HQL file compiling all the required steps sequentially and run it in one go.

1. In your Hive client, log in to the required Hive database.
2. Register the JAR file of Spectrum™ Data & Address Quality for Big Data SDK AMM Module.

```
ADD JAR <Directory path>/amm.hive.${project.version}.jar;
```

3. Create an alias for the Hive UDF of the Data Quality job you wish to run.

   **Note:** String in quotes represents the class names needed for this job to run.

   For example:

```
CREATE TEMPORARY FUNCTION bestofbreed as
'com.pb.bdq.amm.process.hive.consolidation.bestofbreed.BestOfBreedUDAF';
```

4. Switch off the aggregation of data between Reducer and Mapper, by setting the `Hive.Map.Aggr` environment variable configuration to `false`, as indicated in this example:

```
set hive.map.aggr = false;
```

   **Note:** This configuration is required for all UDFs.

5. Specify the configurations and other details for the job, and assign these to respective variables or configuration properties.

   **Note:** The rule must be in JSON format.

   For example,

```
set hivevar:rule='{"consolidationConditions":
[{"consolidationRule":{"conditionClass":"simpleRule",
```

```
"operation":"HIGHEST", "fieldName":"column2", "value":null,
"valueFromField":false, "valueNumeric":true},
"actions":[]}], "removeDuplicates":true}';
```

**Note:** Use the configuration properties in the respective job configurations. For example, `pb.bdq.match.rule`, `pb.bdq.match.express.column`, and `pb.bdq.consolidation.sort.field` where indicated in the respective sample HQL files.

6. Specify the header fields of the input table in comma-separated format, and assign to a variable or configuration property.

```
set hivevar:header ='column1,column2,column3,column4,column5,id';
```

**Note:** Use the configuration property, where indicated. For example, `pb.bdq.match.header`, `pb.bdq.consolidation.header`, and so on where indicated in the respective sample HQL files.

7. Set the sorting parameter to the alias used in the query with the help of the configuration property `'hivevar:sortfield'`.

```
set hivevar:sortfield='id';
```

8. To run the job and display the job output on the console, write the query as indicated in this example:

```
SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
 SELECT  filter (${hivevar:rule},
   ${hivevar:sortfield},
   ${hivevar:header},
   innerRowID.column1,
   innerRowID.column2,
   innerRowID.column3,
   innerRowID.column4,
   innerRowID.column5,
   innerRowID.id
 ) AS matchgroup
 FROM (
  SELECT column1, column2, column3, column4, column5, rowid(*)
  AS id
  FROM data
  ) innerRowID
 GROUP BY column3
```

```
 ) AS innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;
```

To run the job and dump the job output in a designated file, write the query as indicated in this example:

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/HiveUDF/filter/'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
collection items terminated by '||' map keys terminated by ':'
SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
 SELECT  filter (innerRowID.column1,
   innerRowID.column2,
   innerRowID.column3,
   innerRowID.column4,
   innerRowID.column5,
   innerRowID.id
 ) AS matchgroup
 FROM (
  SELECT column1, column2, column3, column4, column5, rowid(*)
  AS id
  FROM data
  ) innerRowID
 GROUP BY column3
 ) AS innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;
```

**Note:** Use the alias defined earlier for the UDF.

# Best of Breed

Best of Breed consolidates duplicate records by selecting the best data in a duplicate record collection and creating a new consolidated record using the best data. This "super" record is known as the best of breed record. You define the rules to use in selecting records to process. When processing completes, the best of breed record is retained by the system.

## Sample Hive Script

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;
```

```
-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.

CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- This rowid is needed by Best of Breed to maintain the order of rows
 while creating groups. This is a UDF (User Defined Function) and
associates an incremental unique integer number to each row of the data.

CREATE TEMPORARY FUNCTION bestofbreed as
'com.pb.bdq.amm.process.hive.consolidation.bestofbreed.BestOfBreedUDAF';
-- Best of Breed is implemented as a UDAF (User Defined Aggregation
function). It processes one group of rows at a time and generates the
result for that group of rows.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'hivevar:rule'

set hivevar:rule='{"consolidationConditions":[
{"consolidationRule":{"conditionClass":"conjoinedRule", "joinType":"AND",
"consolidationRules":[{"conditionClass":"simpleRule",
"operation":"LONGEST", "fieldName":"c5", "value":null,
"valueNumeric":true, "valueFromField":false},
{"conditionClass":"simpleRule", "operation":"IS_NOT_EMPTY",
"fieldName":"c9", "value":null, "valueNumeric":false,
"valueFromField":false}]},
"actions":[{"accumulate":false, "copyFromField":true, "sourceData":"c2",
 "destinationFieldName":"c2"},
{"accumulate":false, "copyFromField":false, "sourceData":"Admin",
"destinationFieldName":"c4"}]},
{"consolidationRule":{"conditionClass":"conjoinedRule", "joinType":"AND",
"consolidationRules":[{"conditionClass":"simpleRule",
"operation":"LONGEST", "fieldName":"c5", "value":null,
"valueNumeric":true, "valueFromField":false},
{"conditionClass":"simpleRule", "operation":"IS_NOT_EMPTY",
"fieldName":"c9", "value":null, "valueNumeric":false,
"valueFromField":false}]},
"actions":[{"accumulate":false, "copyFromField":false,
"sourceData":"Changed", "destinationFieldName":"c10"},
{"accumulate":false, "copyFromField":true, "sourceData":"c5",
"destinationFieldName":"c6"},
{"accumulate":true, "copyFromField":true, "sourceData":"c10",
"destinationFieldName":"c10"}]}],
"keepOriginalRecords":true, "buildTemplateRecord":true,
"templateRules":[{"consolidationRule":{"conditionClass":"conjoinedRule",
 "joinType":"OR",
"consolidationRules":[{"conditionClass":"simpleRule",
"operation":"CONTAINS", "fieldName":"c1", "value":"li",
"valueNumeric":false, "valueFromField":false},
{"conditionClass":"simpleRule", "operation":"LONGEST", "fieldName":"c5",
```

```
 "value":null, "valueNumeric":false, "valueFromField":false}]},
"actions":[]}]}';

-- Set header (along with the id field alias used in the query) using
configuration property 'hivevar:header'
set hivevar:header='c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,id';

-- Set sort field name to the alias used in the query, using the
configuration property 'hivevar:sortField'
set hivevar:sortField='id';

-- Execute Query on the desired table. The query uses a UDF rowid,
which must be present in the query to maintain the ordering of the data
 while reading.
-- Best of Breed returns a list of map containing <key=value> pairs.
Each map in
the list corresponds to a row in the group. The below query explodes
that list of map and fetches fields from map by keys.

SELECT tmp2.record["c1"],
 tmp2.record["c2"],
 tmp2.record["c3"],
 tmp2.record["c4"],
 tmp2.record["c5"],
 tmp2.record["c6"],
 tmp2.record["c7"],
 tmp2.record["c8"],
 tmp2.record["c9"],
 tmp2.record["c10"],
 tmp2.record["CollectionRecordType"]
FROM (
 SELECT  bestofbreed(${hivevar:rule},
  ${hivevar:sortField},
  ${hivevar:header},
  innerRowID.c1,
  innerRowID.c2,
  innerRowID.c3,
  innerRowID.c4,
  innerRowID.c5,
  innerRowID.c6,
  innerRowID.c7,
  innerRowID.c8,
  innerRowID.c9,
  innerRowID.c10,
  innerRowID.id) AS matchgroup
 FROM(
  SELECT c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, rowid(*) AS id FROM
databob
  ) innerRowID
 GROUP BY c3
 ) AS innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;
```

```
-- Query to dump the output to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/bestofbreed/'
 ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' collection items
terminated by '||' map keys terminated by ':'
SELECT tmp2.record["c1"],
 tmp2.record["c2"],
 tmp2.record["c3"],
 tmp2.record["c4"],
 tmp2.record["c5"],
 tmp2.record["c6"],
 tmp2.record["c7"],
 tmp2.record["c8"],
 tmp2.record["c9"],
 tmp2.record["c10"],
 tmp2.record["CollectionRecordType"]
FROM (
 SELECT bestofbreed(innerRowID.c1,
  innerRowID.c2,
  innerRowID.c3,
  innerRowID.c4,
  innerRowID.c5,
  innerRowID.c6,
  innerRowID.c7,
  innerRowID.c8,
  innerRowID.c9,
  innerRowID.c10,
  innerRowID.id) as matchgroup
 FROM(
  SELECT c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, rowid(*) AS id FROM
databob
  ) innerRowID
 GROUP BY c3
 ) AS innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;


--sample input data
--+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+
--| c1       |      c2  |      c3  |      c4  |      c5  |      c6  |
     c7  |      c8  | c9       | c10      |
--+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+
--| Duplicate| 87       | 1        |          |ANNA ABNEY| ANNA     |
          | ABNEY    | A        | 18       |
--| Duplicate| 77       | 1        |          |ANNA A ANN| ANDREA   |
          | ANNAKAY  | A        | 196      |
--+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+

--sample output data
--+------+------+------+------+------+------+------+------+------+------+-------------+
--| c1       | c2       | c3       | c4       | c5       | c6       |
c7       | c8       | c9       | c10      |CollectionRecordType|
--+------+------+------+------+------+------+------+------+------+------+-------------+
```

```
--| Duplicate| 87        | 1         |           |ANNA ABNEY| ANNA     |
         | ABNEY    | A         | 18        |   Primary           |
--| Duplicate| 77        | 1         |           |ANNA A ANN| ANDREA   |
 ARANOW  | ANNAKAY  | A         | 196       |   Secondary         |
--| Duplicate| 87        | 1         |           |ANNA ABNEY| ANNA     |
 ARANOW  | ABNEY    | A         | 18        |   BestOfBreed       |
-+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------------+
```

# Duplicate Synchronization

Duplicate Synchronization determines which fields from a collection of records to copy to the corresponding fields of all records in the collection. You can specify the rules that records must satisfy in order to copy the field data to the other records in the collection. When processing has been completed, all records in the collection are retained.

## Sample Hive Script

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.

CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- Duplicate Sync is implemented as a UDAF (User Defined Aggregation
function).
 It processes one group of rows at a time and generates the result for
 that group of rows.

CREATE TEMPORARY FUNCTION dupsync as
'com.pb.bdq.amm.process.hive.consolidation.duplicatesync.DuplicateSyncUDAF';

-- This rowid is needed by duplicateSync to maintain the order of rows
 while creating groups. This is a UDF (User Defined Function) and
associates
 an incremental unique integer number to each row of the data.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'hivevar:rule'

set hivevar:rule='{"consolidationConditions": [{"consolidationRule":
{"conditionClass":"conjoinedRule", "joinType":"AND",
"consolidationRules":
```

```
[{"conditionClass":"simpleRule", "operation":"HIGHEST",
"fieldName":"column2", "value":null,
 "valueFromField":false, "valueNumeric":true}]},
"actions":[{"accumulate":false, "copyFromField":true,
"sourceData":"column5",
 "destinationFieldName":"column5"}]}]}';

-- Set header (along with the id field alias used in the query)
 using configuration property 'hivevar:header'
set hivevar:header='column1,column2,column3,column4,column5,id';

-- Set sort field name to alias used in query using
configuration property 'hivevar:sortfield'
set hivevar:sortfield='id';

-- Execute Query on the desired table. The query uses a UDF rowid,
which must be present in the query to maintain the ordering of the data
 while reading.
-- Duplicate Sync returns a list of map containing <key=value> pairs.
 Each map in the list corresponds to a row in the group. The below
 query explodes that list of map and fetches fields from map by keys.

SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
 SELECT  dupsync (${hivevar:rule},
  ${hivevar:sortfield},
  ${hivevar:header},
  innerRowID.column1,
  innerRowID.column2,
  innerRowID.column3,
  innerRowID.column4,
  innerRowID.column5,
  innerRowID.id
  ) AS matchgroup
 FROM (
  SELECT column1, column2, column3, column4, column5, rowid(*)
  AS id
  FROM databob
  ) innerRowID
 GROUP BY column3
 ) AS innerResult
 LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;


-- Query to dump the output to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/dupsync/' ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',' collection items terminated by '||'
 map keys terminated by ':'
```

```
SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
 SELECT dupsync( innerRowID.column1,
  innerRowID.column2,
  innerRowID.column3,
  innerRowID.column4,
  innerRowID.column5,
  innerRowID.id
  ) AS matchgroup
 FROM (
  SELECT column1, column2, column3, column4, column5, rowid(*)
  AS id
  FROM databob
  ) innerRowID
 GROUP BY column3 ) AS innerResult
 LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;



--sample input data
--+----------+----------+----------+----------+----------+
--| column1  | column2  | column3  | column4  | column5  |
--+----------+----------+----------+----------+----------+
--| Duplicate| 87       | 1        |          |ANNA ABNEY|
--| Duplicate| 77       | 1        |          |ANNA A ANN|
--| Suspect  |          | 1        |          |ANNA A ABN|
--+----------+----------+----------+----------+----------+

--sample output data
--+----------+----------+----------+----------+----------+
--| column1  | column2  | column3  | column4  | column5  |
--+----------+----------+----------+----------+----------+
--| Duplicate| 87       | 1        |          |ANNA ABNEY|
--| Duplicate| 77       | 1        |          |ANNA A ANN|
--| Suspect  |          | 1        |          |ANNA ABNEY|
--+----------+----------+----------+----------+----------+
```

## Filter

The Filter stage retains or removes records from a group of records based on the rules you specify.

## Sample Hive Script

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.

CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- Filter is implemented as a UDAF (User Defined Aggregation function).

 It processes one group of rows at a time based on join
field and generates the result for that group of rows.

CREATE TEMPORARY FUNCTION filter as
'com.pb.bdq.amm.process.hive.consolidation.filter.FilterUDAF';

-- This rowid is needed by filter to maintain the order of
 rows while creating groups. This is a UDF (User Defined Function)
 and associates an incremental unique integer number to each row of the
 data.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'hivevar:rule'
set hivevar:rule='{"consolidationConditions": [{"consolidationRule":
{"conditionClass":"simpleRule", "operation":"HIGHEST", "fieldName":
"column2", "value":null, "valueFromField":false, "valueNumeric":true},

"actions":[]}], "removeDuplicates":true}';

-- Set header (along with the id field alias used in the query)
 using configuration property 'hivevar:header'
set hivevar:header='column1,column2,column3,column4,column5,id';

-- Set sort field name to alias used in query using
 configuration property 'hivevar:sortfield'
set hivevar:sortfield='id';

-- Execute Query on the desired table. The query uses a UDF rowid,
 which must be present in the query to maintain the ordering of the data
 while reading.

SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
```

```
 SELECT  filter (${hivevar:rule},
   ${hivevar:sortfield},
   ${hivevar:header},
   innerRowID.column1,
   innerRowID.column2,
   innerRowID.column3,
   innerRowID.column4,
   innerRowID.column5,
   innerRowID.id
 ) AS matchgroup
 FROM (
  SELECT column1, column2, column3, column4, column5, rowid(*)
  AS id
  FROM data
  ) innerRowID
 GROUP BY column3
 ) AS innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;

-- Query to dump the output to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/HiveUDF/filter/'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
collection items terminated by '||' map keys terminated by ':'
SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
 SELECT  filter (innerRowID.column1,
   innerRowID.column2,
   innerRowID.column3,
   innerRowID.column4,
   innerRowID.column5,
   innerRowID.id
 ) AS matchgroup
 FROM (
  SELECT column1, column2, column3, column4, column5, rowid(*)
  AS id
  FROM data
  ) innerRowID
 GROUP BY column3
 ) AS innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;


--sample input data
--+----------+----------+----------+----------+----------+
--| column1  | column2  | column3  | column4  | column5  |
--+----------+----------+----------+----------+----------+
--| Duplicate| 80       | 98       |          | EUNICE L |
```

```
--| Suspect  |          | 98       |          | ERIC L BR|
--+----------+----------+----------+----------+----------+

--sample output data
--+----------+----------+----------+----------+----------+
--| column1  | column2  | column3  | column4  | column5  |
--+----------+----------+----------+----------+----------+
--| Suspect  |          | 98       |          | ERIC L BR|
--+----------+----------+----------+----------+----------+
```

## Interflow Match

Interflow Match locates matches between similar data records across two input record streams. The first record stream is a source for suspect records and the second stream is a source for candidate records.

Using match group criteria (for example a match key), Interflow Match identifies a group of records that are potentially duplicates of a particular suspect record.

### *Reporting*

The Interflow Match job allows you to monitor the results of the job. The counters available are:

| | |
|---|---|
| **DUPLICATE_COLLECTIONS** | The number of duplicate collections, which consist of a suspect and its duplicate records grouped together by a CollectionNumber. |
| **EXPRESS_MATCHES** | The number of Express Matches made in a collection.<br>An Express Match is made when a suspect and candidate have an exact match on the contents of a designated field, usually an ExpressMatchKey provided by the Match Key Generator. If an Express Match is made, no further processing is done to determine if the suspect and candidate are duplicates. |
| **AVERAGE_SCORE** | The average match score of all duplicates.<br>The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match. |
| **INPUT_SUSPECTS** | The number of records in the input stream that the matcher tried to match to other records. |
| **SUSPECTS_WITH_DUPLICATES** | The number of input suspects that matched at least one candidate record. |
| **UNIQUE_SUSPECTS** | The number of input suspects that did not match any candidate records. |

| | |
|---|---|
| **SUSPECTS_WITH_CANDIDATES** | The number of input suspects that had at least one candidate record in its match group and therefore had at least one match attempt. |
| **SUSPECTS_WITHOUT_CANDIDATES** | The number of input suspects that had no candidate records in its match group and therefore had no match attempts. |
| **TOTAL_DUPLICATE_CANDIDATES** | The total number of duplicate candidates found. |
| **TOTAL_DUPLICATE_SCORE** | The total match score of all the duplicates. |

### Sample Hive Script

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.
CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- This rowid is needed by Interflow Match to maintain the
order of rows while creating groups. This is a UDF (User Defined
Function)
and associates an incremental unique integer number to each row of the
 data.

CREATE TEMPORARY FUNCTION InterMatch as
'com.pb.bdq.amm.process.hive.interflow.InterMatchUDAF';

-- Inter Flow is implemented as a UDAF (User Defined Aggregation
function).
 It processes one group of rows at a time based on join field
 and generates the result for that group of rows.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'hivevar:rule'

set hivevar:rule='{"type":"Parent", "missingDataMethod":
"IgnoreBlanks", "threshold":100.0, "weight":0,
"children":[{"type":"Child", "missingDataMethod":"IgnoreBlanks",
"threshold":80.0,
 "weight":0, "matchWhenNotTrue":false, "scoringMethod":"Maximum",
"algorithms":[{"name":"EditDistance", "weight":0, "options":null},
{"name":"Metaphone", "weight":0, "options":null}],
"crossMatchField":[], "suspectField":"firstname", "candidateField":null},
{"type":"Child", "missingDataMethod":"IgnoreBlanks",
 "threshold":80.0, "weight":0,
"matchWhenNotTrue":false, "scoringMethod":"Maximum",
"algorithms":[{"name":"KeyboardDistance",
```

```
"weight":0, "options":null},
{"name":"Metaphone3", "weight":0,
 "options":null}], "crossMatchField":[],
"suspectField":"lastname", "candidateField":null}],
"scoringMethod":"Average", "matchingMethod":"AllTrue",
 "name":"NameData", "matchWhenNotTrue":false}';


-- Set the header for suspect table using configuration property
'hivevar:suspectheader'
set
hivevar:suspectheader='name,firstname,lastname,matchkey,middlename,recordid';

-- Set the header for candidate table using configuration property
'hivevar:Candidateheader'
set
hivevar:Candidateheader='name,firstname,lastname,matchkey,middlename,recordid';

-- Set the sorting field to the candidates unique id's
 alias used in the query. This is not from the input data.
set hivevar:sortfield='c_id';

-- Set the express match column(optional)
set hivevar:expressMatchColumn='matchkey';


-- Optionally, one can also set
'hivevar:intercomparison='returnUniqueCandidates,true'';
set hivevar:intercomparison='returnUniqueCandidates,true';

-- Set sort collection number option for unique records using
 configuration property 'hivevar:collectionNumberZero'
set hivevar:collectionNumberZero='false';

-- Execute Query on the desired table. The query uses a UDF rowid,
which must be present in the query to maintain the ordering of the data
 while reading.

SELECT innerresult.record ["MatchRecordType"],
 innerresult.record ["MatchScore"],
 innerresult.record ["HasDuplicate"],
 innerresult.record ["CollectionNumber"],
 coalesce(innerresult.record ["ExpressMatched"], ''),
 innerresult.record ["SourceType"],
 innerresult.record ["name"],
 innerresult.record ["firstname"],
 innerresult.record ["lastname"],
 innerresult.record ["matchkey"],
 innerresult.record ["middlename"],
 innerresult.record ["recordid"]
FROM (
 SELECT
interMatch(${hivevar:rule},${hivevar:sortfield},${hivevar:expressMatchColumn},
```

```
${hivevar:collectionNumberZero},${hivevar:interComparison},${hivevar:Candidateheader}


${hivevar:Suspectheader},unionresults.id,unionresults.name,unionresults.firstname,

        unionresults.lastname, unionresults.matchkey,
unionresults.middlename,
        unionresults.recordid ,unionresults.TYPE)
       AS matchgroup
    FROM (
 SELECT rowid(*) AS id, 'Suspect' AS TYPE,fullname as name,fname
    as firstname,lname as lastname,matchkey as matchkey,mname as
middlename,recordid as recordid
    FROM customer_name_suspect
    UNION ALL
 SELECT rowid(*) AS id , 'Candidate' AS TYPE, name as name,firstname as

 firstname,lastname as lastname,matchkey as matchkey,middlename as
middlename ,customerid as recordid
    FROM customer_name_candidate) unionresults
    GROUP BY matchkey) AS innerResult LATERAL VIEW
explode(innerResult.matchgroup) innerresult AS record;



-- Query to dump data to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/intermatch/output'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
collection items terminated by '||' map keys terminated by ':'
SELECT innerresult.record ["MatchRecordType"],
 innerresult.record ["MatchScore"],
 innerresult.record ["HasDuplicate"],
 innerresult.record ["CollectionNumber"],
 coalesce(innerresult.record ["ExpressMatched"], ''),
 innerresult.record ["SourceType"],
 innerresult.record ["name"],
 innerresult.record ["firstname"],
 innerresult.record ["lastname"],
 innerresult.record ["matchkey"],
 innerresult.record ["middlename"],
 innerresult.record ["recordid"]
FROM (
 SELECT
interMatch(${hivevar:rule},${hivevar:sortfield},${hivevar:expressMatchColumn},

       ${hivevar:collectionNumberZero},${hivevar:interComparison},
       ${hivevar:Candidateheader},${hivevar:Suspectheader},
       unionresults.id,unionresults.name,unionresults.firstname,
       unionresults.lastname, unionresults.matchkey,
```

```
      unionresults.middlename,unionresults.recordid ,unionresults.TYPE)
 AS matchgroup
    FROM (
 SELECT rowid(*) AS id, 'Suspect' AS TYPE,fullname as name,fname as
       firstname,lname as lastname,matchkey as matchkey,mname as
middlename,recordid as recordid
    FROM customer_name_suspect
    UNION ALL
 SELECT rowid(*) AS id , 'Candidate' AS TYPE, name as name,
    firstname as firstname,lastname as lastname,matchkey as matchkey,
    middlename as middlename ,customerid as recordid
    FROM customer_name_candidate) unionresults
    GROUP BY matchkey) AS innerResult LATERAL VIEW
explode(innerResult.matchgroup) innerresult AS record;


-- Sample input Suspect data

--+----------------+---------+---------------+------------+-----------+----------+
--| name             | firstname| lastname        | matchkey    |
middlename | recordid  |
--+----------------+---------+---------------+------------+-----------+----------+
--| LAURA  ABADSANTOS| LAURA    | ABADSANTOS      | L           |
       | 1         |
--+----------------+---------+---------------+------------+-----------+----------+

-- Sample input candidate data

--+----------------+---------+---------------+------------+-----------+----------+
--| name             | firstname| lastname        | matchkey    |
middlename | recordid  |
--+----------------+---------+---------------+------------+-----------+----------+
--| KATHRYN E ABATE  | KATHRYN  | ABATE           | L           |  E
       | 3         |
--| ANNA  ABAYEV     | ANNA     | ABAYEV          | L           |
       | 5         |
--+----------------+---------+---------------+------------+-----------+----------+

-- Sample output data

-+--------+-----+------+---------+--------+-----+------+-----+----+----+-----+------+
--|MatchRecordType|MatchScore|HasDuplicate|CollectionNumber|ExpressMatched|SourceType|
 name       | firstname| lastname|matchkey|middlename| recordid  |
-+--------+-----+------+---------+--------+-----+------+-----+----+----+-----+------+
--|S              |0         |Y           |0-0-1           |
    |S         |LAURA ABADSA| LAURA    |ABADSANTO|    L   |          |
 1         |
--|D              |80        |D           |0-0-1           |N
    |C         |KATHRYN E AB| KATHRYN  |AB       |    L   |    E     |
 3         |
--|D              |90        |D           |0-0-1           |N
    |C         |ANNA ABAYEV | ANNA     |ABAYEV   |    L   |          |
 5         |
```

```
+----------+--------+---------+----------+---------+--------+---------+--------+------+--------+---------+
```

# Intraflow Match

Intraflow Match locates matches between similar data records within a single input stream. You can create hierarchical rules based on any fields that have been defined or created in other stages of the dataflow.

*Reporting*

The Intraflow Match job allows you to monitor the results of the job. The counters available are:

| | |
|---|---|
| **INPUT_RECORDS** | The number of records in the matching stage before the matching sort is performed. |
| **DUPLICATE_RECORDS** | The number of duplicate records within a match group, which can be either a suspect or a candidate record. |
| **UNIQUE_RECORDS** | The number of suspect or candidate records which do not match any other records in their respective match group.<br>If it is the only record in a match group, a suspect is automatically unique. |
| **MATCH_GROUPS** | (Group By) Records grouped together by a match key. |
| **DUPLICATE_COLLECTIONS** | The number of duplicate collections, which consist of a suspect and its duplicate records grouped together by a CollectionNumber. |
| **EXPRESS_MATCHES** | The number of Express Matches made in a collection.<br>An Express Match is made when a suspect and candidate have an exact match on the contents of a designated field, usually an ExpressMatchKey provided by the Match Key Generator. If an Express Match is made, no further processing is done to determine if the suspect and candidate are duplicates. |
| **AVERAGE_SCORE** | The average match score of all duplicates.<br>The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match. |
| **TOTAL_DUPLICATES** | The total number of duplicates found. |
| **TOTAL_SCORE** | The total match score of all duplicates. |

### Sample Hive Script

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;
```

```
-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.
CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- This rowid is needed by Intraflow Match to maintain
 the order of rows while creating groups. This is a UDF (User Defined
Function)
and associates an incremental unique integer number to each row of the
 data.

CREATE TEMPORARY FUNCTION intraMatch as
'com.pb.bdq.amm.process.hive.intraflow.IntraMatchUDAF';
-- Intra Flow is implemented as a UDAF (User Defined Aggregation
function).
 It processes one group of rows at a time and generates the result for
 that group of rows

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'hivevar:rule'
set hivevar:rule='{"type":"Parent",
"children":[{"type":"Child", "matchWhenNotTrue":false, "threshold":80.0,
 "weight":0,
"algorithms":[{"name":"EditDistance", "weight":0, "options":null},
{"name":"Metaphone", "weight":0, "options":null}],
"scoringMethod":"Maximum", "missingDataMethod":"IgnoreBlanks",
 "crossMatchField":[], "suspectField":"firstname",
"candidateField":null},
{"type":"Child", "matchWhenNotTrue":false, "threshold":80.0, "weight":0,

"algorithms":[{"name":"KeyboardDistance", "weight":0, "options":null},
{"name":"Metaphone3", "weight":0, "options":null}],
"scoringMethod":"Maximum",
"missingDataMethod":"IgnoreBlanks", "crossMatchField":[],
"suspectField":"lastname", "candidateField":null}],
"matchingMethod":"AllTrue", "scoringMethod":"Average",
"missingDataMethod":"IgnoreBlanks",
"name":"NameData", "matchWhenNotTrue":false,
"threshold":100,"weight":0}';


-- Set header(along with id field alias used in query) using
configuration property 'hivevar:header'
set hivevar:header='firstname,lastname,matchkey,middlename,id';

-- Set the express match column (optional)
set hivevar:expresscolumn='matchkey';

-- Set sort field name to the alias used in the query,
 using the configuration property 'hivevar:sortfield'
```

```
set hivevar:sortfield='id';

-- Set sort collection number option for unique records using
configuration property 'hivevar:UniqueCollectionNumber'
set hivevar:UniqueCollectionNumber='false';

-- Execute Query on the desired table. The query uses a UDF rowid,
which must be present in the query to maintain the ordering of the data
 while reading.
-- Intra Match returns a list of map containing <key=value> pairs.
Each map in the list corresponds to a row in the group.
The below query explodes that list of map and fetches fields from map
by keys.

SELECT innerresult.record["MatchRecordType"],
 innerresult.record["MatchScore"],
 innerresult.record["CollectionNumber"],
 innerresult.record["ExpressMatched"],
 innerresult.record["firstname"],
 innerresult.record["lastname"],
 innerresult.record["matchkey"],
 innerresult.record["middlename"]
 FROM (
  SELECT  intraMatch( ${hivevar:rule},
    ${hivevar:sortfield},
    ${hivevar:expresscolumn},
    ${hivevar:UniqueCollectionNumber},
    ${hivevar:header},
    innerRowID.firstname,
    innerRowID.lastname,
    innerRowID.matchkey,
    innerRowID.middlename,
    innerRowID.id
  ) AS matchgroup
  FROM (
   SELECT  firstname, lastname, matchkey, middlename, rowid(*)
   AS id
   FROM customer_data
   ) innerRowID
  GROUP BY matchkey
 ) AS innerResult
 LATERAL VIEW explode(innerResult.matchgroup) innerresult AS record ;

-- Query to dump output to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/IntraFlow/'
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' collection items terminated
 by '||' map keys terminated by ':'
SELECT innerresult.record["MatchRecordType"],
 innerresult.record["MatchScore"],
 innerresult.record["CollectionNumber"],
 innerresult.record["ExpressMatched"],
 innerresult.record["firstname"],
```

```
 innerresult.record["lastname"],
 innerresult.record["matchkey"],
 innerresult.record["middlename"]
 FROM (
  SELECT  intraMatch(innerRowID.firstname,
    innerRowID.lastname,
    innerRowID.matchkey,
    innerRowID.middlename,
    innerRowID.id
  ) AS matchgroup
  FROM (
    SELECT  firstname, lastname, matchkey, middlename, rowid(*)
    AS id
    FROM customer_data
   ) innerRowID
  GROUP BY matchkey
  ) AS innerResult
 LATERAL VIEW explode(innerResult.matchgroup) innerresult AS record ;


--sample input data
--+-------------+-------------+---------------+--------------+
--| firstname   | lastname    | middlename    | matchkey     |
--+-------------+-------------+---------------+--------------+
--| Steven      | Aaen        | LYRIC         | AAE          |
--| DEBRA       | AALMO       | BOATMAN       | AAE          |

--| MARY        | AARON       | ROLLING MEADOW| AAE          |
--+-------------+-------------+---------------+--------------+

--sample output data
—+————————+———————+————————+———————+———————————+———————————+———————————+————————+
--| firstname | lastname|middlename  |
matchkey|MatchRecordType|CollectionNumber|ExpressMatched|MatchScore|
—+————————+———————+————————+———————+———————————+———————————+———————————+————————+
--| Steven    | Aaen    | LYRIC      | AAE     |    S        |
 0-0-1  |     Y        |     0    |
--| DEBRA     | AALMO   | BOATMAN    | AAE     |    D        |
 0-0-1  |     Y        |    100   |
--| MARY      | AARON   | ROLLING MEA| AAE     |    D        |
 0-0-1  |     Y        |    100   |
—+————————+———————+————————+———————+———————————+———————————+———————————+————————+
```

## Match Key Generator

Match Key Generator creates a non-unique key for each record, which can then be used by matching stages to identify groups of potentially duplicate records. Match keys facilitate the matching process by allowing you to group records by match key and then only comparing records within these groups.

The match key is created using rules you define and is comprised of input fields. Each input field specified has a selected algorithm that is performed on it. The result of each algorithm is then concatenated to create a single match key field.

In addition to creating match keys, you can also create express match keys to be used later in the dataflow by an Intraflow Match stage or an Interflow Match stage.

You can create multiple match keys and express match keys.

For example, if the incoming record is:

First Name - Fred
Last Name - Mertz
Postal Code - 21114-1687
Gender Code - M

And you define a match key rule that generates a match key by combining data from the record like this:

| Input Field | Start Position | Length |
|---|---|---|
| Postal Code | 1 | 5 |
| Postal Code | 7 | 4 |
| Last Name | 1 | 5 |
| First Name | 1 | 5 |
| Gender Code | 1 | 1 |

Then the key would be:

211141687MertzFredM

### Sample Hive Script

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.
CREATE TEMPORARY FUNCTION matchkeygenerator as
 'com.pb.bdq.amm.process.hive.matchkeygenerator.MatchKeyGeneratorUDF';

-- Match Key Generator is implemented as a UDF (User Defined function).
```

```
 It processes one row at a time and generates a map of match keys for
each row.

-- Set rule and header
set
hivevar:rule='{"matchKeys":[{"expressMatchKey":false,"matchKeyField":"MatchKey1",


"rules":[{"algorithm":"Soundex","field":"busniessname","startPosition":1,"length":0,


"active":true,"sortInput":null,"removeNoiseCharacters":false}]},
                {"expressMatchKey":false,"matchKeyField":"MatchKey2",

"rules":[{"algorithm":"Koeln","field":"busniessname","startPosition":1,"length":0,


"active":true,"sortInput":null,"removeNoiseCharacters":false}]}]}';


set hivevar:header='busniessname,recordid';

-- Execute query on the desired table to display the job output on
console.
 This query returns a map of key value for each row containing matchkeys
 as per rule passed.
SELECT busniessname, recordid, bar.ret["MatchKey1"] AS MatchKey1,
bar.ret["MatchKey2"]
AS MatchKey2 FROM (SELECT *, matchkeygenerator (${hivevar:rule},
${hivevar:header},
 busniessname, recordid) AS ret FROM cust ) bar;


-- Query to dump output to a directory in file system
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/MatchKey/' row format
 delimited FIELDS TERMINATED BY ',' MAP FIELDS TERMINATED BY ':'
COLLECTION ITEMS TERMINATED
BY '|' LINES TERMINATED BY '\n' STORED AS TEXTFILE
SELECT busniessname, recordid, bar.ret["MatchKey1"] AS MatchKey1,
 bar.ret["MatchKey2"] AS MatchKey2 FROM (
SELECT *, matchkeygenerator (${hiveconf:rule}, ${hiveconf:header},
 busniessname, recordid) AS ret FROM cust ) bar;

--Sample data in input table customer
--+-----------------------------------------+----------------+--+
--|            cust.busniessname            | cust.recordid  |
--+-----------------------------------------+----------------+--+
--| Internal Revenue Service                | 0              |
--| Juan F Vera-Monroig                     | 1              |
--| Leonardo Pagan-Reyes                    | 2              |
--| Academia San Joaquin Colegios/Academias | 3              |
--| Nereida Portalatin-Padua                | 4              |
--+-----------------------------------------+----------------+--+
```

```
--Sample output for input query
+-------------------------------------+----------+-----------+-----------------+--+
|               busniessname          | recordid  | matchkey1  |
    matchkey2     |
+-------------------------------------+----------+-----------+-----------------+--+
| Internal Revenue Service            | 0         | I536       |
0627657368738     |
| Juan F Vera-Monroig                 | 1         | J511       |
063376674         |
| Leonardo Pagan-Reyes                | 2         | L563       |
567214678         |
| Academia San Joaquin Colegios/Academias  | 3         | A235       |
0426864645484268  |
| Nereida Portalatin-Padua            | 4         | N631       |
67217252612       |
+-------------------------------------+----------+-----------+-----------------+--+
```

## Transactional Match

Transactional Match matches suspect records against candidate records of a group of records to identify duplicates. The records are first grouped by a selected column, post which the first record is marked as the suspect record. All the remaining records of the group, termed as candidate records, are matched against the suspect record.

If the candidate record is a duplicate, it is assigned a collection number, the match record type is labeled a Duplicate, and the record is then written out. Any unmatched candidates in the group are assigned a collection number of 0, labeled as Unique and then written out as well.

### *Reporting*

The Transactional Match job allows you to monitor the results of the job. The counters available are:

| | |
|---|---|
| **AVERAGE_SCORE** | The average match score of all duplicates.<br>The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match. |
| **INPUT_SUSPECTS** | The number of records in the input stream that the matcher tried to match to other records. |
| **SUSPECTS_WITH_DUPLICATES** | The number of input suspects that matched at least one candidate record. |
| **UNIQUE_SUSPECTS** | The number of input suspects that did not match any candidate records. |

| | |
|---|---|
| **SUSPECTS_WITH_CANDIDATES** | The number of input suspects that had at least one candidate record in its match group and therefore had at least one match attempt. |
| **SUSPECTS_WITHOUT_CANDIDATES** | The number of input suspects that had no candidate records in its match group and therefore had no match attempts. |
| **TOTAL_DUPLICATES_SCORE** | The total match score of all duplicates. |
| **TOTAL_DUPLICATES** | The total number of duplicates found. |

### Sample Hive Script

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.

CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- This rowid is needed by Transactional Match to maintain the
order of rows while creating groups. This is a UDF (User Defined
Function) and
associates an incremental unique integer number to each row of the data.

CREATE TEMPORARY FUNCTION transactionalMatch as
 'com.pb.bdq.amm.process.hive.transactional.TransactionalMatchUDAF';

-- Transactional Match is implemented as a UDAF (User Defined Aggregation
 function).
It processes one group of rows at a time and generates the result for
that group of rows.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'hivevar:rule'
set hivevar:rule='{"type":"Parent", "children":[{"type":"Child",
"matchWhenNotTrue":false,
 "threshold":80.0, "weight":0, "algorithms":[{"name":"EditDistance",
"weight":0, "options":null},
{"name":"Metaphone","weight":0,"options":null}],
"scoringMethod":"Maximum", "missingDataMethod":"IgnoreBlanks",
"crossMatchField":[],
 "suspectField":"firstname", "candidateField":null},
{"type":"Child", "matchWhenNotTrue":false, "threshold":80.0,
 "weight":0,
"algorithms":[{"name":"KeyboardDistance","weight":0,"options":null},
{"name":"Metaphone3","weight":0,"options":null}],
"scoringMethod":"Maximum", "missingDataMethod":"IgnoreBlanks",
```

```
"crossMatchField":[], "suspectField":"lastname", "candidateField":null}],

 "matchingMethod":"AllTrue", "scoringMethod":"Average",
"missingDataMethod":"IgnoreBlanks",
 "name":"NameData", "matchWhenNotTrue":false, "threshold":100,
"weight":0}';

-- Set header(along with id field alias used in query) using
configuration property 'hivevar:header'
set
hivevar:header='name,firstname,lastname,matchkey,middlename,recordid,id';

-- Set sort field name to the alias used in the query,
 using the configuration property 'hivevar:sort'
set hivevar:sort='id';

-- Set sort collection number option for unique records using
configuration property
 'hivevar:uniquecolumnsreturn'. The default value is false.
set hivevar:uniquecolumnsreturn='true';

-- Execute Query on the desired table. The query uses a UDF rowid,
 which must be present in the query to maintain the ordering of the data
 while reading.
-- Transactional Match returns a list of map containing <key=value>
pairs.
 Each map in the list corresponds to a row in the group. The below query

explodes that list of map and fetches fields from map by keys.

SELECT tmp2.record["MatchRecordType"],
 tmp2.record["MatchScore"],
 tmp2.record["HasDuplicate"],
 tmp2.record["name"],
 tmp2.record["firstname"],
 tmp2.record["lastname"],
 tmp2.record["matchkey"],
 tmp2.record["middlename"],
 tmp2.record["recordid"]
FROM (
 SELECT  transactionalMatch(innerRowID.name, innerRowID.firstname,
 innerRowID.lastname, innerRowID.matchkey, innerRowID.middlename,
 innerRowID.recordid, innerRowID.id
 ) AS matchgroup
 FROM (
  SELECT name, firstname, lastname, matchkey, middlename,
              recordid, rowid(name, firstname, lastname, matchkey,
middlename,
              recordid) AS id FROM customer_data
  ) innerRowID
 GROUP BY matchkey
) As innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 as record ;
```

```
-- Query to dump output to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/transmatch/' ROW FORMAT
 DELIMITED
 FIELDS TERMINATED BY ',' collection items terminated by '||' map keys
 terminated by ':'
SELECT tmp2.record["MatchRecordType"],
 tmp2.record["MatchScore"],
 tmp2.record["HasDuplicate"],
 tmp2.record["name"],
 tmp2.record["firstname"],
 tmp2.record["lastname"],
 tmp2.record["matchkey"],
 tmp2.record["middlename"],
 tmp2.record["recordid"]
FROM (
 SELECT  transactionalMatch(${hivevar:rule},
   ${hivevar:sort},
   ${hivevar:uniquecolumnsreturn},
   ${hivevar:header},
   innerRowID.name,
   innerRowID.firstname,
   innerRowID.lastname,
   innerRowID.matchkey,
   innerRowID.middlename,
   innerRowID.recordid,
   innerRowID.id) as matchgroup
 FROM (
  SELECT name, firstname, lastname, matchkey, middlename,
recordid, rowid(name, firstname, lastname, matchkey, middlename,
recordid) AS id
  FROM customer_data
  ) innerRowID
 GROUP BY matchkey ) As innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 as record ;



--sample input data
-+----------+-------------+-------------+-------------+--------------+-------------+-+
--| name               | firstname              | lastname
 | matchkey              | middlename              | recordid
     |
-+----------+-------------+-------------+-------------+--------------+-------------+-+
--| ZORINA  ABDOOL     | ZORINA                 | ABDOOL
 | Z                     |                         | 12
     |
--| ZULFIQAR  ALI      | ZULFIQAR               | ALI
 | Z                     |                         | 116
     |
--| ZACHARY  BENNETT   | ZACHARY                | BENNETT
```

```
 | Z                     |                          | 515
     |
--| ZOHAR  BUERGER     | ZOHAR                    | BUERGER
 | Z                     |                          | 889
     |
-+-----------+-----------+-----------+-----------+-----------+-------+-+

--sample output data
-+-----------+-----------+-----------+-----------+-----------+-------+-+
--|name            |firstname | lastname | matchkey | middlename |
recordid  | MatchRecordType | MatchScore | HasDuplicate |
-+-----------+-----------+-----------+-----------+-----------+-------+-+
--|ZORINA  ABDOOL  |ZORINA    | ABDOOL   | Z        |            | 12
       |      S    | 0    | Y      |
--|ZULFIQAR  ALI   |ZULFIQAR  | ALI      | Z        |            | 116
       |      D          | 90   | D      |
--|ZACHARY  BENNETT|ZACHARY   | BENNETT  | Z        |            | 515
       |      D          | 91   | D      |
--|ZOHAR  BUERGER  |ZOHAR     | BUERGER  | Z        |            | 889
       |      D    | 91   | D      |
-+-----------+-----------+-----------+-----------+-----------+-------+-+
```

# Data Integration Module Functions

## Using a Custom Groovy Script Hive UDF

To run each Hive UDF job, you can either run these steps individually on your Hive client within a single session, or create an HQL file compiling all the required steps sequentially and run it in one go.

1. In your Hive client, log in to the required Hive database.
2. Register the JAR file of Spectrum™ Data & Address Quality for Big Data SDK DIM Module.

```
ADD JAR /home/hduser/script/dim.hive.${project.version}.jar;
```

3. Create an alias for the Hive UDF of the `CustomGroovyScript` job.

**Note:** String in quotes represents the class names needed for this job to run.

For example,

```
CREATE TEMPORARY FUNCTION customscript as
'com.pb.bdq.dim.process.hive.script.groovy.CustomGroovyScriptExecutionUDF';
```

4. Enable or disable the hive fetch task conversion.
   For Example,

```
set hive.fetch.task.conversion=none;
```

5. Use `hivevar:defaultConfiguration` to specify the date, date-time, and time pattern. Assign this configuration to the respective variable.

```
set hivevar:defaultConfiguration='{"datePattern":"M/d/yy",
"dateTimePattern":"M/d/yy h:mm a","timePattern":"h:mm a"}';
```

   **Note:** This is an optional configuration.

6. Specify the header fields of the input table in comma-separated format, and assign to a variable.
   For example,

```
set hivevar:header='busniessname,recordid';
```

7. Use `hivevar:scriptConfigurations` to set the groovy script configurations. It includes details, such as groovyScriptFile, inputFields, and outputFields
   For Example,

```
set hivevar:scriptConfigurations =
'[{"groovyScriptFile":"/home/hduser/script/groovy_hive.txt",
"inputFields":[{"name":"busniessname","type":"string"},
{"name":"recordid","type":"integer"}],
"outputFields":[{"name":"outtan","type":"double"}]},
{"groovyScriptFile":"/home/hduser/script/groovy2.txt",
"inputFields":[],
"outputFields":[{"name":"outtan2","type":"double"}]}]';
```

8. To run the job and display the job output on the console, write the query as indicated in this example:

   **Note:** This query returns output fields as transformed by the groovy script.

```
SELECT customscript(${hivevar:scriptConfigurations},"",
${hivevar:header}, InputKeyValue, AddressLine1) FROM groovy_tc1;
```

To run the job and dump the output in a designated file, write the query as indicated in this example:

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hduser/script/output'
row format delimited FIELDS TERMINATED BY ',' lines TERMINATED BY '\n'

STORED AS TEXTFILE SELECT * FROM (SELECT customscript
(${hivevar:scriptConfigurations},
${hivevar:defaultConfiguration},${hivevar:header},
InputKeyValue, AddressLine1)
as mygp FROM groovy_tc1 ) record;
!q;
```

**Note:** Use the alias defined earlier for the UDF.

## Sample Hive Script

```
-- Register Data Integration Module [DIM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dim.hive.${project.version}.jar;

ADD FILE /home/hduser/script/groovy_hive.txt;
ADD FILE /home/hduser/script/groovy2.txt;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION customscript as
'com.pb.bdq.dim.process.hive.script.groovy.CustomGroovyScriptExecutionUDF';

set hive.fetch.task.conversion=none;

-- Set default configuration
set hivevar:defaultConfiguration='{"datePattern":"M/d/yy",
"dateTimePattern":"M/d/yy h:mm a","timePattern":"h:mm a"}';

-- Set the header configuration
set hivevar:header='busniessname,recordid';

set hivevar:scriptConfigurations =
'[{"groovyScriptFile":"/home/hduser/script/groovy_hive.txt",
"inputFields":[{"name":"busniessname","type":"string"},
{"name":"recordid","type":"integer"}],
"outputFields":[{"name":"outtan","type":"double"}]},
{"groovyScriptFile":"/home/hduser/script/groovy2.txt",
"inputFields":[],
"outputFields":[{"name":"outtan2","type":"double"}]}]';

SELECT customscript(${hivevar:scriptConfigurations},"",${hivevar:header},
InputKeyValue, AddressLine1) FROM groovy_tc1;
```

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hduser/script/output'
row format delimited FIELDS TERMINATED BY ',' lines TERMINATED BY '\n'

STORED AS TEXTFILE SELECT * FROM (SELECT
customscript(${hivevar:scriptConfigurations},
${hivevar:defaultConfiguration},${hivevar:header}, InputKeyValue,
AddressLine1)
as mygp FROM groovy_tc1 ) record;
!q;
```

# Data Normalization Module Functions

## Using a Hive UDF of Data Normalization Module

To run each Hive UDF job, you can either run these steps individually on your Hive client within a single session, or create an HQL file compiling all the required steps sequentially and run it in one go.

1. In your Hive client, log in to the required Hive database.
2. Register the JAR file of Spectrum™ Data & Address Quality for Big Data SDK DNM Module.

   ```
   ADD JAR <Directory path>/dnm.hive.${project.version}.jar;
   ```

3. Create an alias for the Hive UDF of the Data Quality job you wish to run.

   **Note:** String in quotes represents the class names needed for this job to run.

   For example:

   ```
   CREATE TEMPORARY FUNCTION tablelookup as
   'com.pb.bdq.dnm.process.hive.tablelookup.TableLookUpUDF';
   ```

4. Specify the reference data path.
   - **Reference data is on HDFS**
     - Reference data is to be **downloaded to a working directory** for jobs

- If the reference data is in *unarchived* file format, set the reference directory as:

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"./referenceData","dataDownloader":{"dataDownloader":"DC"}}';
```

- If the reference data is in *archived* format, set the reference directory as:

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"./referenceData.zip","dataDownloader":
{"dataDownloader":"DC"}}';
```

- Reference data is to be **downloaded on local nodes** for jobs. In this case, set the reference data directory as:

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"/home/data/dm/referenceData","dataDownloader":{"dataDownloader":
"HDFS","localFSRepository":"/local/download"}}';
```

- **Reference data is on local path**: Ensure that data is present on each node of the cluster on the same path.

  Set the reference directory as:

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"LocaltoDataNodes",
"dataDir":"/home/data/referenceData"}';
```

5. Specify the configurations and other details for the job, and assign these to respective variables or configuration properties.

   **Note:** The rule must be in JSON format.

   For example,

```
set hivevar:rule='{"rules":[{"action":"Standardize",
"source":"CityCode", "tableName":"State Name Abbreviations",
"lookupMultipleWordTerms":false,
"lookupIndividualTermsWithinField":false,
"destination":"CityCode"}]}';
```

   **Note:** Ensure to use the configuration properties in the respective job configurations. For example, `pb.bdq.match.rule`, `pb.bdq.match.express.column`, `pb.bdq.consolidation.sort.field`, and so on where indicated in the respective sample HQL files.

6. Specify the header fields of the input table in comma-separated format, and assign to a variable or configuration property.

```
set hivevar:header
='AccountDescription,Address,ApartmentNumber,CityCode';
```

7. To run the job and display the job output on the console, write the query as indicated in this example:

```
SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hivevar:rule}, ${hiveconf:refdir},
${hiveconf:header}, accountdescription, address, apartmentnumber,
citycode)
 AS ret
 FROM citizen_data
 ) bar;
```

To run the job and dump the job output in a designated file, write the query as indicated in this example:

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/TableLookup/' row format
 delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED
AS TEXTFILE
SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hiveconf:rule}, ${hiveconf:refdir},
${hiveconf:header}, accountdescription, address, apartmentnumber,
citycode)
 AS ret
 FROM citizen_data
 ) bar;
```

**Note:** Use the alias defined earlier for the UDF.

# Advanced Transformer

The Advanced Transformer job scans and splits strings of data into multiple fields using tables or regular expressions. It extracts a specific term or a specified number of words to the right or left of a term. Extracted and non-extracted data can be placed into an existing field or a new field.

For example, want to extract the suite information from this address field and place it in a separate field.

2300 BIRCH RD STE 100

To accomplish this, you could create an Advanced Transformer that extracts the term STE and all words to the right of the term STE, leaving the field as:

2300 BIRCH RD

## Sample Hive Script

### *Reference data placed on local node*

```
-- Register Data Normalisation Module [DNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes
represent class names needed for this job to run.
-- Advanced Transformer is implemented as a UDF(User Defined function).

Hence it processes one row at a time and generates a map of key value
pairs for each row.
CREATE TEMPORARY FUNCTION advanceTransform
as
'com.pb.bdq.dnm.process.hive.advancedtransformer.AdvancedTransformerUDF';

-- Set rule
set hivevar:rule='{"rules":[{"extractionType":"TableData",
"source":"address", "nonExtractedData":"address_1",
"extractedData":"address_2",
 "tokenizationCharacters":"", "tableName":"Street Suffix Abbreviations",
 "multipleTermLookup":false,
"tokenize":true, "extract":"ExtractTerm",
"includeTermWith":"ExtractedData", "wordsToExtract":2}]}';

-- Set Reference Directory. This must be a local path on cluster machines
 and must
 be present on each node of the cluster at the same path.
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"LocaltoDataNodes",
"dataDir":"/home/data/referenceData"}';
```

```
-- set header
set hivevar:header ='AccountDescription,Address';

set hive.fetch.task.conversion=none;

-- Execute Query on the desired table, to display the job output
on console. This query returns a map of key value pairs containing output
 fields for each row.

SELECT bar.ret["AdvancedTransformTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["address_1"]
FROM (
 SELECT advanceTransform(${hivevar:rule}, ${hivevar:refereceDataDetails},

 ${hivevar:header}, accountdescription, address)
 AS ret
 FROM advxformX
 ) bar;


-- Query to dump output data to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/AdvXformer/' row format

delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS
TEXTFILE
SELECT bar.ret["AdvancedTransformTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["address_1"]
FROM (
 SELECT advanceTransform(${hivevar:rule}, ${hivevar:refereceDataDetails},

${hivevar:header}, accountdescription, address)
 AS ret
 FROM advxformX
 ) bar;

--sample input data
+---------------------------------+--------------------+----------------------+
| AdvancedTransformTermIdentified  | accountdescription  |      address
          |
+---------------------------------+--------------------+----------------------+
| Yes                              |                    | 400 E M0 St
 Apt 1405  |
| Yes                              |                    | 190 E 72nd
St         |
+---------------------------------+--------------------+----------------------+

--sample output data
+--------------------------+----------------+-----------------+----------------+
```

```
| AdvancedTransformTermIdentified  | accountdescription  |      address
          |     address_1      |
+----------------------------------+---------------------+-----------------+
| Yes                              |                     | 400 E M0 St
 Apt 1405  | 400 E M0 Apt 1405  |
| Yes                              |                     | 190 E 72nd
St          | 190 E 72nd         |
+----------------------------------+---------------------+-----------------+
```

*Reference data placed on HDFS and downloaded on local nodes for jobs*

```
-- Register Data Normalisation Module [DNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional).
 String in quotes represent class names needed for this job to run.
-- Advanced Transformer is implemented as a UDF(User Defined function).

 Hence it processes one row at a time and generates a map of key value
 pairs for each row.
CREATE TEMPORARY FUNCTION advanceTransform as

'com.pb.bdq.dnm.process.hive.advancedtransformer.AdvancedTransformerUDF';


-- Set rule
set hivevar:rule='{"rules":[{"extractionType":"TableData",
"source":"address",
 "nonExtractedData":"address_1", "extractedData":"address_2",
"tokenizationCharacters":"",
 "tableName":"Street Suffix Abbreviations", "multipleTermLookup":false,
 "tokenize":true,
 "extract":"ExtractTerm", "includeTermWith":"ExtractedData",
"wordsToExtract":2}]}';


-- Set reference data details for Download manager, paas dataDir where
 data resides in HDFS
 and localFS path to download the data and dataDownloader as HDFS
set hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"/home/data/dm/referenceData",
"dataDownloader":{"dataDownloader":"HDFS","localFSRepository":"/local/download"}}';

set hive.fetch.task.conversion=none;

-- set header
set hivevar:header ='AccountDescription,Address';

set hive.fetch.task.conversion=none;

-- Execute Query on the desired table, to display the job output on
console. This query returns a map of key value pairs containing output
```

```
 fields for each row.

SELECT bar.ret["AdvancedTransformTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["address_1"]
FROM (
 SELECT advanceTransform(${hivevar:rule}, ${hivevar:refereceDataDetails},

 ${hivevar:header}, accountdescription, address)
 AS ret
 FROM advxformX
 ) bar;


-- Query to dump output data to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/AdvXformer/'
row format delimited FIELDS TERMINATED BY ',' lines terminated by '\n'
 STORED AS TEXTFILE
SELECT bar.ret["AdvancedTransformTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["address_1"]
FROM (
 SELECT advanceTransform(${hivevar:rule}, ${hivevar:refereceDataDetails},

 ${hivevar:header}, accountdescription, address)
 AS ret
 FROM advxformX
 ) bar;

--sample input data
+---------------------------------+--------------------+----------------------+
| AdvancedTransformTermIdentified  | accountdescription  |      address
          |
+---------------------------------+--------------------+----------------------+
| Yes                              |         | 400 E M0 St Apt 1405  |
| Yes                              |                     | 190 E 72nd
St         |
+---------------------------------+--------------------+----------------------+

--sample output data
+-------------------------+-----------------+-----------------+----------------+
| AdvancedTransformTermIdentified  | accountdescription  |      address
          |     address_1      |
+-------------------------+-----------------+-----------------+----------------+
| Yes                              |         | 400 E M0 St Apt 1405  |
 400 E M0 Apt 1405  |
| Yes                              |                     | 190 E 72nd
St         | 190 E 72nd         |
+-------------------------+-----------------+-----------------+----------------+
```

*Reference data placed on HDFS and downloaded to working directory for jobs*

```
-- Register Data Normalisation Module [DNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional).
 String in quotes represent class names needed for this job to run.
-- Advanced Transformer is implemented as a UDF(User Defined function).

Hence it processes one row at a time and generates a map of key value
pairs for each row.
CREATE TEMPORARY FUNCTION advanceTransform as
'com.pb.bdq.dnm.process.hive.advancedtransformer.AdvancedTransformerUDF';


-- HDFS Reference data- via Distributed Cache DC- unarchive
ADD FILES hdfs://<HOST>:<PORT>/home/hduser/referenceData/;

--HDFS reference data- via Distributed cache -archive form of reference
 data
--ADD ARCHIVE hdfs://<HOST>:<PORT>/home/hduser/referenceData.zip;

-- Set rule
set hivevar:rule='{"rules":[{"extractionType":"TableData",
 "source":"address", "nonExtractedData":"address_1",
"extractedData":"address_2",
 "tokenizationCharacters":"", "tableName":"Street Suffix Abbreviations",
 "multipleTermLookup":false,
 "tokenize":true, "extract":"ExtractTerm",
"includeTermWith":"ExtractedData", "wordsToExtract":2}]}';


-- Set Reference Data details .

--reference data details, can be added in zip or unarchive form, dataDir
 symbolises reference data
set hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"./referenceData","dataDownloader":{"dataDownloader":"DC"}}';

-- below format for archive form
--set hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"./referenceData.zip","dataDownloader":{"dataDownloader":"DC"}}';


-- set header
set hivevar:header ='AccountDescription,Address';

set hive.fetch.task.conversion=none;

-- Execute Query on the desired table, to display the job output on
console.
This query returns a map of key value pairs containing output fields
for each row.
```

```
SELECT bar.ret["AdvancedTransformTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["address_1"]
FROM (
 SELECT advanceTransform(${hivevar:rule}, ${hivevar:refereceDataDetails},

       ${hivevar:header}, accountdescription, address)
 AS ret
 FROM advxformX
 ) bar;


-- Query to dump output data to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/AdvXformer/'
row format delimited FIELDS TERMINATED BY ',' lines terminated by '\n'
 STORED AS TEXTFILE
SELECT bar.ret["AdvancedTransformTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["address_1"]
FROM (
 SELECT advanceTransform(${hivevar:rule}, ${hivevar:refereceDataDetails},

       ${hivevar:header}, accountdescription, address)
 AS ret
 FROM advxformX
 ) bar;

--sample input data
+---------------------------------+--------------------+----------------------+
| AdvancedTransformTermIdentified  | accountdescription  |      address
         |
+---------------------------------+--------------------+----------------------+
| Yes                              |                    | 400 E M0 St
 Apt 1405  |
| Yes                              |                    | 190 E 72nd
St         |
+---------------------------------+--------------------+----------------------+

--sample output data
+---------------------------+-----------------+-----------------+----------------+
| AdvancedTransformTermIdentified  | accountdescription  |      address
         |     address_1      |
+---------------------------+-----------------+-----------------+----------------+
| Yes                              |         | 400 E M0 St Apt 1405  |
 400 E M0 Apt 1405  |
| Yes                              |                    | 190 E 72nd
St         | 190 E 72nd         |
+---------------------------+-----------------+-----------------+----------------+
```

# Open Parser

Open Parser parses your input data from many cultures of the world using a simple but powerful parsing grammar. Using this grammar, you can define a sequence of expressions that represent domain patterns for parsing your input data. Open Parser also collects statistical data and scores the parsing matches to help you determine the effectiveness of your parsing grammars.

Use Open Name Parser to:

- Parse input data using domain-specific and culture-specific parsing grammars that you define in Domain Editor.
- Parse input data using domain-independent parsing grammars that you define in Open Parser using the same simple but powerful parsing grammar available in Domain Editor.
- Parse input data using domain-independent parsing grammars at runtime that you define in Dataflow Options.
- Preview parsing grammars to test how sample input data parses before running the job using the target input data file.
- Trace parsing grammar results to view how tokens matched or did not match the expressions you defined and to better understand the matching process.

## Sample Hive Script

### *Reference data placed on local node*

```
-- Register Data Normalisation Module [DNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Register the grammar for open parser
ADD FILE
/home/hadoop/testing/openparser/hive/testdata/Open_parser_grammer_tc1.txt;

-- Provide alias to UDF class (optional). String in quotes
represent class names needed for this job to run.
-- Open Parser is implemented as a UDF(User Defined function).
 Hence it processes one row at a time and generates a map of key value
 pairs for each row.

CREATE TEMPORARY FUNCTION openparser as
'com.pb.bdq.dnm.process.hive.openparser.OpenParserUDF';

set hive.fetch.task.conversion=minimal;

set hivevar:header='InputKeyValue,addressline';

-- Set Reference Directory. This must be a local path on
cluster machines and must be present on each node of the cluster at the
 same path.
```

```
set hivevar:refereceDataDetails='{"referenceDataPathLocation":
"LocaltoDataNodes","dataDir":"/home/data/referenceData"}';

-- Set rule with grammar file name
set hivevar:rule='{"grammar":null,"domain":null,"defaultCulture":null,
"culture":null,"returnMultipleParsedRecords":false,"debug":false,
"grammerFilePath":"Open_parser_grammer_tc1.txt"}';

--Execute Query on desired table to sump the output to local dir.
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/testing/openparser/hive/testdata/output'
row format delimited FIELDS TERMINATED BY '|'
lines terminated by '\n'
STORED AS TEXTFILE
SELECT coalesce(tmp2.record ["InputKeyValue"], '')
 ,coalesce(tmp2.record ["addressline"], '')
 ,coalesce(tmp2.record ["ApartmentNumber"], '')
 ,coalesce(tmp2.record ["HouseNumber"], '')
 ,coalesce(tmp2.record ["IsParsed"], '')
 ,coalesce(tmp2.record ["LeadingDirectional"], '')
 ,coalesce(tmp2.record ["ParserScore"], '')
 ,coalesce(tmp2.record ["POBox"], '')
 ,coalesce(tmp2.record ["PrivateMailbox"], '')
 ,coalesce(tmp2.record ["RRHC"], '')
 ,coalesce(tmp2.record ["RuleID"], '')
 ,coalesce(tmp2.record ["StreetName"], '')
 ,coalesce(tmp2.record ["StreetSuffix"], '')
 ,coalesce(tmp2.record ["TrailingDirectional"], '')
FROM (
 SELECT openparser(${hivevar:rule},${hivevar:refereceDataDetails},
       ${hivevar:header}, InputKeyValue, addressline) AS mygp
 FROM openparserinputtable1
 ) AS addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 AS
record;
```

### *Reference data placed on HDFS and downloaded on local nodes for jobs*

```
-- Register Data Normalisation Module [DNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Register the grammar for open parser
ADD FILE
/home/hadoop/testing/openparser/hive/testdata/Open_parser_grammer_tc1.txt;

-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.
-- Open Parser is implemented as a UDF(User Defined function).
Hence it processes one row at a time and generates a map of key value
pairs for each row.

CREATE TEMPORARY FUNCTION openparser as
'com.pb.bdq.dnm.process.hive.openparser.OpenParserUDF';
```

```
set hive.fetch.task.conversion=minimal;

set hivevar:header='InputKeyValue,addressline';

-- Set reference data details for Download manager, paas
dataDir where data resides in HDFS and localFS path to download the data
 and dataDownloader as HDFS
set hivevar:refereceDataDetails='{"referenceDataPathLocation":
"HDFS","dataDir":"/home/data/dm/referenceData","dataDownloader":
{"dataDownloader":"HDFS","localFSRepository":"/local/download"}}';

set hive.fetch.task.conversion=none;

-- Set rule with grammar file name
set hivevar:rule='{"grammar":null,"domain":null,"defaultCulture":null,
"culture":null,"returnMultipleParsedRecords":false,"debug":false,
"grammerFilePath":"Open_parser_grammer_tc1.txt"}';

--Execute Query on desired table to sump the output to local dir.
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/testing/openparser/hive/testdata/output'
row format delimited FIELDS TERMINATED BY '|'
lines terminated by '\n'
STORED AS TEXTFILE
SELECT coalesce(tmp2.record ["InputKeyValue"], '')
 ,coalesce(tmp2.record ["addressline"], '')
 ,coalesce(tmp2.record ["ApartmentNumber"], '')
 ,coalesce(tmp2.record ["HouseNumber"], '')
 ,coalesce(tmp2.record ["IsParsed"], '')
 ,coalesce(tmp2.record ["LeadingDirectional"], '')
 ,coalesce(tmp2.record ["ParserScore"], '')
 ,coalesce(tmp2.record ["POBox"], '')
 ,coalesce(tmp2.record ["PrivateMailbox"], '')
 ,coalesce(tmp2.record ["RRHC"], '')
 ,coalesce(tmp2.record ["RuleID"], '')
 ,coalesce(tmp2.record ["StreetName"], '')
 ,coalesce(tmp2.record ["StreetSuffix"], '')
 ,coalesce(tmp2.record ["TrailingDirectional"], '')
FROM (
 SELECT openparser(${hivevar:rule},${hivevar:refereceDataDetails},
       ${hivevar:header}, InputKeyValue, addressline) AS mygp
 FROM openparserinputtable1
 ) AS addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 AS
record;
```

*Reference data placed on HDFS and downloaded to working directory for jobs*

```
-- Register Data Normalisation Module [DNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;
```

```
-- Register the grammar for open parser
ADD FILE
/home/hadoop/testing/openparser/hive/testdata/Open_parser_grammer_tc1.txt;


-- HDFS Reference data- via Distributed Cache DC- unarchive
ADD FILES hdfs://<HOST>:<PORT>/home/hduser/referenceData/;

--HDFS reference data- via Distributed cache -archive form of reference
 data
--ADD ARCHIVE hdfs://<HOST>:<PORT>/home/hduser/referenceData.zip;

-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.
-- Open Parser is implemented as a UDF(User Defined function).
 Hence it processes one row at a time and generates a map of key value
 pairs for each row.

CREATE TEMPORARY FUNCTION openparser as
 'com.pb.bdq.dnm.process.hive.openparser.OpenParserUDF';

set hive.fetch.task.conversion=minimal;

set hivevar:header='InputKeyValue,addressline';


-- Set Reference Data details .

--reference data details, can be added in zip or
 unarchive form, dataDir symbolises reference data
set hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"./referenceData","dataDownloader":{"dataDownloader":"DC"}}';

-- below format for archive form
--set hivevar:refereceDataDetails='{"referenceDataPathLocation":
"HDFS","dataDir":"./referenceData.zip","dataDownloader":{"dataDownloader":"DC"}}';

-- Set rule with grammar file name
set hivevar:rule='{"grammar":null,"domain":null,"defaultCulture":null,
"culture":null,"returnMultipleParsedRecords":false,"debug":
false,"grammerFilePath":"Open_parser_grammer_tc1.txt"}';

--Execute Query on desired table to sump the output to local dir.
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/testing/openparser/hive/testdata/output'
row format delimited FIELDS TERMINATED BY '|'
lines terminated by '\n'
STORED AS TEXTFILE
SELECT coalesce(tmp2.record ["InputKeyValue"], '')
 ,coalesce(tmp2.record ["addressline"], '')
 ,coalesce(tmp2.record ["ApartmentNumber"], '')
 ,coalesce(tmp2.record ["HouseNumber"], '')
```

```
 ,coalesce(tmp2.record ["IsParsed"], '')
 ,coalesce(tmp2.record ["LeadingDirectional"], '')
 ,coalesce(tmp2.record ["ParserScore"], '')
 ,coalesce(tmp2.record ["POBox"], '')
 ,coalesce(tmp2.record ["PrivateMailbox"], '')
 ,coalesce(tmp2.record ["RRHC"], '')
 ,coalesce(tmp2.record ["RuleID"], '')
 ,coalesce(tmp2.record ["StreetName"], '')
 ,coalesce(tmp2.record ["StreetSuffix"], '')
 ,coalesce(tmp2.record ["TrailingDirectional"], '')
FROM (
 SELECT openparser(${hivevar:rule},${hivevar:refereceDataDetails},
       ${hivevar:header}, InputKeyValue, addressline) AS mygp
 FROM openparserinputtable1
 ) AS addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 AS
record;
```

# Table Lookup

The Table Lookup stage standardizes terms against a previously validated form of that term and applies the standard version. This evaluation is done by searching a table for the term to standardize.

### Sample Hive Script

#### *Reference data placed on local node*

```
-- Register Data Normalization Modue [dnm] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.
-- Table Lookup is implemented as a UDF (User Defined function).
 Hence it processes one row at a time and generates a map of key value
 pairs for each row.
CREATE TEMPORARY FUNCTION tablelookup as
'com.pb.bdq.dnm.process.hive.tablelookup.TableLookUpUDF';

-- Set rule
set hivevar:rule='{"rules":[{"action":"Standardize", "source":"CityCode",

 "tableName":"State Name Abbreviations", "lookupMultipleWordTerms":false,

 "lookupIndividualTermsWithinField":false, "destination":"CityCode"}]}';

-- Set Reference Directory. This must be a local path on cluster machines

and must be present on each node of the cluster at the same path.
set
```

```
hivevar:refereceDataDetails='{"referenceDataPathLocation":"LocaltoDataNodes",
"dataDir":"/home/data/referenceData"}';

set hive.fetch.task.conversion=none;

-- set header
set hivevar:header
='AccountDescription,Address,ApartmentNumber,CityCode';


-- Execute Query on the desired table, to display the job output on
console.
 This query returns a map of key value pairs containing output fields
for each row.

SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hivevar:rule}, ${hivevar:refereceDataDetails},
${hivevar:header},
       accountdescription, address, apartmentnumber, citycode)
 AS ret
 FROM citizen_data
 ) bar;


-- Query to dump output data to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/TableLookup/' row format

 delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS
 TEXTFILE
SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hiveconf:rule}, ${hiveconf:refereceDataDetails},

      ${hiveconf:header}, accountdescription, address, apartmentnumber,
 citycode)
 AS ret
 FROM citizen_data
 ) bar;

--Sample input data
--+-------------------------+---------------+----------------------+-----------------+
--| citizen_data.accountdescription  | citizen_data.address  |
citizen_data.apartmentnumber  | citizen_data.citycode |
```

```
--+-----------------------+---------------+---------------------+-----------------+
--|           | 400 E M0 St Apt 1405  |                                     |
 NY                       |
--|                                   | 190 E 72nd St          |
                         | NY                        |
--|                                   | 1381 3rd Ave Apt 4     | 4
                         | TTYYY                     |
--+-----------------------+---------------+---------------------+-----------------+


--sample output data
--+---------------------+-----------------+---------------+-----------------+-------+

--|StandardizationTermIdentified |       accountdescription | address
               | apartmentnumber       | citycode|
--+---------------------+-----------------+---------------+-----------------+-------+

--| yes         |               | 400 E M0 St Apt 1405  |
        | NEW YORK |
--| yes         |                             | 190 E 72nd St          |
                      | NEW YORK |
--| yes         |                             | 1381 3rd Ave Apt 4     | 4
                      | NEW YORK |
--+---------------------+-----------------+---------------+-----------------+-------+
```

### *Reference data placed on HDFS and downloaded on local nodes for jobs*

```
-- Register Data Normalization Modue [dnm] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.
-- Table Lookup is implemented as a UDF (User Defined function).
Hence it processes one row at a time and generates a map of key value
pairs for each row.
CREATE TEMPORARY FUNCTION tablelookup as
'com.pb.bdq.dnm.process.hive.tablelookup.TableLookUpUDF';

-- Set rule
set hivevar:rule='{"rules":[{"action":"Standardize", "source":"CityCode",

 "tableName":"State Name Abbreviations", "lookupMultipleWordTerms":false,

"lookupIndividualTermsWithinField":false, "destination":"CityCode"}]}';

-- Set reference data details for Download manager, paas dataDir where

data resides in HDFS and localFS path to download the data and
dataDownloader as HDFS
set hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
```

```
"dataDir":"/home/data/dm/referenceData","dataDownloader":{"dataDownloader":"HDFS",
"localFSRepository":"/local/download"}}';

set hive.fetch.task.conversion=none;

-- set header
set hivevar:header
='AccountDescription,Address,ApartmentNumber,CityCode';


-- Execute Query on the desired table, to display the job output on
console.
This query returns a map of key value pairs containing output fields
for each row.

SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hivevar:rule}, ${hivevar:refereceDataDetails},
 ${hivevar:header}, accountdescription, address, apartmentnumber,
citycode)
 AS ret
 FROM citizen_data
 ) bar;


-- Query to dump output data to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/TableLookup/'
row format delimited FIELDS TERMINATED BY ',' lines terminated by '\n'
 STORED AS TEXTFILE
SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hiveconf:rule}, ${hiveconf:refereceDataDetails},
 ${hiveconf:header}, accountdescription, address, apartmentnumber,
citycode)
 AS ret
 FROM citizen_data
 ) bar;

--Sample input data
--+-------------------------+---------------+----------------------+----------------+
--| citizen_data.accountdescription  | citizen_data.address  |
citizen_data.apartmentnumber  | citizen_data.citycode |
--+-------------------------+---------------+----------------------+----------------+
--|          | 400 E M0 St Apt 1405  |                                 |
```

```
 NY                        |
--|                                  | 190 E 72nd St          |
                          | NY                        |
--|                                  | 1381 3rd Ave Apt 4     | 4
                          | TTYYY                     |
--+--------------------------+-----------------+-------------------------+-----------------+


--sample output data
--+----------------------+-----------------+---------------+-----------------+-------+

--|StandardizationTermIdentified |       accountdescription | address
        | apartmentnumber       | citycode|
--+----------------------+-----------------+---------------+-----------------+-------+

--|yes          |               | 400 E M0 St Apt 1405  |
        | NEW YORK |
--|yes          |                             | 190 E 72nd St          |
                     | NEW YORK |
--|yes          |                             | 1381 3rd Ave Apt 4     | 4
                     | NEW YORK |
--+----------------------+-----------------+---------------+-----------------+-------+
```

*Reference data placed on HDFS and downloaded to working directory for jobs*

```
-- Register Data Normalization Modue [dnm] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.
-- Table Lookup is implemented as a UDF (User Defined function).
 Hence it processes one row at a time and generates a map of key value
 pairs for each row.
CREATE TEMPORARY FUNCTION tablelookup as
 'com.pb.bdq.dnm.process.hive.tablelookup.TableLookUpUDF';

-- HDFS Reference data- via Distributed Cache DC- unarchive
ADD FILES hdfs://<HOST>:<PORT>/home/hduser/referenceData/;

--HDFS reference data- via Distributed cache -archive form of reference
 data
--ADD ARCHIVE hdfs://<HOST>:<PORT>/home/hduser/referenceData.zip;
-- Set rule
set hivevar:rule='{"rules":[{"action":"Standardize",
"source":"CityCode", "tableName":"State Name Abbreviations",
 "lookupMultipleWordTerms":false,
"lookupIndividualTermsWithinField":false,
"destination":"CityCode"}]}';

-- Set Reference Data details .

--reference data details, can be added in zip
```

```
or unarchive form, dataDir symbolises reference data
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS","dataDir":"./referenceData",
"dataDownloader":{"dataDownloader":"DC"}}';

-- below format for archive form
--set hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"./referenceData.zip","dataDownloader":{"dataDownloader":"DC"}}';

set hive.fetch.task.conversion=none;

-- set header
set hivevar:header
='AccountDescription,Address,ApartmentNumber,CityCode';


-- Execute Query on the desired table, to display the job output on
console. This query returns a map of key value pairs containing output
 fields for each row.

SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hivevar:rule}, ${hivevar:refereceDataDetails},
${hivevar:header}, accountdescription, address, apartmentnumber,
citycode)
 AS ret
 FROM citizen_data
 ) bar;


-- Query to dump output data to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/TableLookup/' row format

 delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS
 TEXTFILE
SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hiveconf:rule}, ${hiveconf:refereceDataDetails},
 ${hiveconf:header}, accountdescription, address, apartmentnumber,
citycode)
 AS ret
 FROM citizen_data
 ) bar;
```

```
--Sample input data
--+-------------------------+-----------------+-------------------------+-----------------+
--| citizen_data.accountdescription  | citizen_data.address  |
citizen_data.apartmentnumber  | citizen_data.citycode |
--+-------------------------+-----------------+-------------------------+-----------------+
--|          | 400 E M0 St Apt 1405  |                                    |
 NY                      |
--|                                  | 190 E 72nd St          |
                        | NY                        |
--|                                  | 1381 3rd Ave Apt 4    | 4
                        | TTYYY                     |
--+-------------------------+-----------------+-------------------------+-----------------+


--sample output data
--+---------------------+-----------------+---------------+---------------+-------+

--|StandardizationTermIdentified | accountdescription       | address
        | apartmentnumber       | citycode|
--+---------------------+-----------------+---------------+---------------+-------+

--|yes          |               | 400 E M0 St Apt 1405  |
        | NEW YORK |
--|yes          |                           | 190 E 72nd St          |
                    | NEW YORK |
--|yes          |                           | 1381 3rd Ave Apt 4    | 4
                    | NEW YORK |
--+---------------------+-----------------+---------------+---------------+-------+
```

# Global Addressing Module Functions

## Using a Hive UDF of Global Addressing Module

To run each Hive UDF job, you can either run these steps individually on your Hive client within a single session, or create an HQL file compiling all the required steps sequentially and run it in one go.

1. In your Hive client, log in to the required Hive database.
2. Register the JAR file of Spectrum™ Data & Address Quality for Big Data SDK GAM Module.

   ```
   ADD JAR /home/hduser/gam/
   gam.globaladdressvalidation.hive.${project.version}.jar
   ```

3. Create an alias for the Hive UDF of the Address Quality job you want to run.

**Note:** String in quotes represents the class names needed for this job to run.

For example:

```
CREATE TEMPORARY FUNCTION addressvalidation as
'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';
```

4. Enable or disable the hive fetch task conversion.
   For Example:

```
set hive.fetch.task.conversion=none;
```

5. Use `hivevar:engineconf` to set the engine configurations, such as, Database Path, Country Code, and the Process Type.
   For Example:

```
set hivevar:engineconf='{"productDatabaseInfoList":
[{"dbPath":"/home/hduser/ReferenceData/AddressQuality/GAM/GGB062017",
"countryCode":"GBR","processType":"VALIDATE"}]}';
```

6. Specify the settings for the input data using the `hivevar:inputoption` parameter.
   For example:

```
set hivevar:inputoption='{"casing":"Mixed","matchMode":"Relaxed",
"defaultCountry":"GBR","maximumResults":2,"returnInputAddress":false,
"returnParsedAddress":false,"returnPrecisionCode":false,"mustMatchAddressNumber":false,
"mustMatchStreet":false,"mustMatchCity":false,"mustMatchLocality":false,
"mustMatchState":false,"mustMatchStateProvince":false,"mustMatchPostCode":false,
"keepMultiMatch":true,"preferPostalOverCity":false,"cityFallback":true,
"postalFallback":true,"validationLevel":"ADDRESS"}';
```

7. Specify the header fields of the input table in comma-separated format, and assign to a variable or configuration property.

```
set
hivevar:header='inputkeyvalue,AddressLine1,AddressLine2,City,postalcode,
StateProvince,firmname,Country';
```

8. To run the job and display the job output on the console, write the query as indicated in this example:

   **Note:** This query returns a map of key value pairs containing output fields for each row.

```
SELECT
tmp2.record["HouseNumber"],tmp2.record["Confidence"],tmp2.record["AddressLine1"],
tmp2.record["StreetName"],tmp2.record["PostalCode"] FROM (SELECT
addressvalidation(${hivevar:engineconf},${hivevar:inputoption},
${hivevar:header},inputkeyvalue,addressline1,AddressLine2,city,postalcode,stateprovince,firmname,
country)as mygp from address_validation) as addressgroup LATERAL VIEW
 explode(addressgroup.mygp)tmp2 as record;
```

   To run the job and dump the job output in a designated file, write the query as indicated in this example:

```
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hduser/GlobalAddressValidation/'row format delimited
FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS TEXTFILE
 SELECT * FROM (SELECT addressvalidation(${hivevar:engineconf},
${hivevar:inputoption},${hivevar:header},inputkeyvalue,addressline1,AddressLine2,city,postalcode,
stateprovince,firmname,country)as mygp from address_validation) as
addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

   **Note:** Use the alias defined earlier for the UDF.

# Global Address Validation

Global Addressing Validation provides enhanced address standardization and validation for international addresses outside the U.S. Global Address Validation is part of the Global Addressing Module.

## Sample Hive Scripts

### *Reference data placed on local node*

```
-- Register Global Addressing Module [Global Address Validation] BDQ
Hive UDF Jar
```

```
ADD JAR <directory
path>/gam-globaladdressvalidation-hive-${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes
 represent class names needed for this job to run.
CREATE TEMPORARY FUNCTION addressvalidation as
'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';

-- set Engine configuration
set hivevar:engineconf='{"productDatabaseInfoList":
[{"dbPath":"<path to extracted spd>",
"countryCode":["FRA"],"processType":"VALIDATE"}],
"referenceDataPathLocation":"LocaltoDataNodes"}';

-- set input configuration
set hivevar:inputoption='{"casing":"Upper","matchMode":"Relaxed",
"defaultCountry":"FRA","maximumResults":1,"returnInputAddress":true,
"returnParsedAddress":true,"returnPrecisionCode":true,
"returnCountrySpecificFields":true,"mustMatchAddressNumber":false,
"mustMatchStreet":false,"mustMatchCity":false,"mustMatchLocality":false,
"mustMatchState":false,"mustMatchStateProvince":false,
"mustMatchPostCode":false,"keepMultiMatch":false,"preferPostalOverCity":false,
"cityFallback":false,"postalFallback":false,"validationLevel":"ADDRESS"}';

-- set header
set hivevar:header='RecordID,AddressLine1,City,PostalCode,Country';

-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '<local directory path>' row format
delimited FIELDS TERMINATED BY '|' lines terminated by '\n' STORED AS
TEXTFILE
SELECT
coalesce(tmp2.record["AdditionalInputData"] ,''),
coalesce(tmp2.record["AddressBlock1"] ,''),
coalesce(tmp2.record["AddressBlock2"] ,''),
coalesce(tmp2.record["AddressLine1"] ,''),
coalesce(tmp2.record["AddressLine1.Input"] ,''),
coalesce(tmp2.record["ApartmentLabel"] ,''),
coalesce(tmp2.record["ApartmentNumber"] ,''),
coalesce(tmp2.record["AUS.Address.Class"] ,''),
coalesce(tmp2.record["AUS.Level.Number"] ,''),
coalesce(tmp2.record["AUS.Parcel.ID"] ,''),
coalesce(tmp2.record["AUS.PID"] ,''),
coalesce(tmp2.record["AUS.Principal.Pid"] ,''),
coalesce(tmp2.record["AUS.SA1"] ,''),
coalesce(tmp2.record["Building"] ,''),
coalesce(tmp2.record["CAN.Census.CD"] ,''),
coalesce(tmp2.record["CAN.Census.CMA"] ,''),
coalesce(tmp2.record["CAN.Census.CSD"] ,''),
coalesce(tmp2.record["CAN.Census.CT"] ,''),
coalesce(tmp2.record["CAN.Census.DA"] ,''),
coalesce(tmp2.record["CAN.FormattedStreet.Range"] ,''),
coalesce(tmp2.record["City"] ,''),
```

```
coalesce(tmp2.record["City.Input"] ,''),
coalesce(tmp2.record["City.Matched"] ,''),
coalesce(tmp2.record["CitySubdivision"] ,''),
coalesce(tmp2.record["CitySubdivision.Input"] ,''),
coalesce(tmp2.record["CitySubdivision.Matched"] ,''),
coalesce(tmp2.record["Confidence"] ,''),
coalesce(tmp2.record["CouldNotValidate"] ,''),
coalesce(tmp2.record["Country"] ,''),
coalesce(tmp2.record["Country.Input"] ,''),
coalesce(tmp2.record["County"] ,''),
coalesce(tmp2.record["County.Matched"] ,''),
coalesce(tmp2.record["FirmName"] ,''),
coalesce(tmp2.record["FirmName.Input"] ,''),
coalesce(tmp2.record["Firmname.Matched"] ,''),
coalesce(tmp2.record["GBR.Aliased.Locality"] ,''),
coalesce(tmp2.record["GBR.Dependent.Locality"] ,''),
coalesce(tmp2.record["GBR.DependentStreet.Name"] ,''),
coalesce(tmp2.record["GBR.DoubleDependent.Locality"] ,''),
coalesce(tmp2.record["GBR.Historic.Postcode"] ,''),
coalesce(tmp2.record["GBR.OSAPR"] ,''),
coalesce(tmp2.record["GBR.RPC"] ,''),
coalesce(tmp2.record["GBR.UPRN"] ,''),
coalesce(tmp2.record["HouseNumber"] ,''),
coalesce(tmp2.record["Housenumber.Matched"] ,''),
coalesce(tmp2.record["IRL.Eircode"] ,''),
coalesce(tmp2.record["ITA.Historical.Postcode"] ,''),
coalesce(tmp2.record["LeadingDirectional"] ,''),
coalesce(tmp2.record["MatchOnAllStreetFields"] ,''),
coalesce(tmp2.record["MatchOnStreetDirectional"] ,''),
coalesce(tmp2.record["MultimatchCount"] ,''),
coalesce(tmp2.record["NZL.Aliased.SUBURB"] ,''),
coalesce(tmp2.record["ParsedAddressLine1.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentLable.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentNumber.Input"] ,''),
coalesce(tmp2.record["ParsedCity.Input"] ,''),
coalesce(tmp2.record["ParsedCitySubDivision.Input"] ,''),
coalesce(tmp2.record["ParsedCountry.Input"] ,''),
coalesce(tmp2.record["ParsedHouseNumber.Input"] ,''),
coalesce(tmp2.record["ParsedPlaceName.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeAddOn.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeBase.Input"] ,''),
coalesce(tmp2.record["ParsedPostStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedPreStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvince.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["PlaceName"] ,''),
coalesce(tmp2.record["POBox"] ,''),
coalesce(tmp2.record["PostalCode"] ,''),
coalesce(tmp2.record["PostalCode.AddOn"] ,''),
coalesce(tmp2.record["PostalCode.Input"] ,''),
coalesce(tmp2.record["Postalcode.Matched"] ,''),
coalesce(tmp2.record["PrecisionCode"] ,''),
coalesce(tmp2.record["Principality"] ,''),
```

```
coalesce(tmp2.record["ProcessedBy"] ,''),
coalesce(tmp2.record["RecordID"] ,''),
coalesce(tmp2.record["StateProvince"] ,''),
coalesce(tmp2.record["StateProvince.Input"] ,''),
coalesce(tmp2.record["StateProvince.Matched"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Matched"] ,''),
coalesce(tmp2.record["Status"] ,''),
coalesce(tmp2.record["Status.Code"] ,''),
coalesce(tmp2.record["Status.Description"] ,''),
coalesce(tmp2.record["StreetName"] ,''),
coalesce(tmp2.record["StreetName.Matched"] ,''),
coalesce(tmp2.record["StreetType"] ,''),
coalesce(tmp2.record["StreetType.Matched"] ,''),
coalesce(tmp2.record["Subcity"] ,''),
coalesce(tmp2.record["TrailingDirectional"] ,''),
coalesce(tmp2.record["VendorCode"] ,'')
 FROM (SELECT  addressvalidation
(${hivevar:engineconf},${hivevar:inputoption},
${hivevar:header},RecordID,AddressLine1,City,PostalCode,Country)
as mygp from gavtable )as addressgroup
 LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

### *Reference data placed on HDFS and downloaded on local nodes for jobs*

```
-- Register Global Addressing Module [Global Address Validation] BDQ
Hive UDF Jar
ADD JAR <directory
path>/gam-globaladdressvalidation-hive-${project.version}.jar;

-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.
CREATE TEMPORARY FUNCTION addressvalidation as
'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';

-- set Engine configuration
set hivevar:engineconf='{"productDatabaseInfoList":[{"referenceDataPath":
{"referenceDataPathLocation":"HDFS",
"dataDir":"/user/hadoop/RefrenceData/AddressValidation",
"dataDownloader":{"dataDownloader":"HDFS","localFSRepository":
"/opt/PitneyBowes/ReferenceData/AddressValidation"}},
"countryCode":["GBR"],"processType":"VALIDATE"}]}';

-- set input configuration
set hivevar:inputoption='{"casing":"Upper","matchMode":
"Relaxed","defaultCountry":"GBR","maximumResults":1,"returnInputAddress"
:true,"returnParsedAddress":true,"returnPrecisionCode":true,"returnCountrySpecificFields":true,
"mustMatchAddressNumber":false,"mustMatchStreet":false,"mustMatchCity":false,
"mustMatchLocality":false,"mustMatchState":false,
"mustMatchStateProvince":false,"mustMatchPostCode":false,"keepMultiMatch":false,
"preferPostalOverCity":false,"cityFallback":false,
```

```
"postalFallback":false,"validationLevel":"ADDRESS"}';

-- set header
set hivevar:header='RecordID,AddressLine1,City,PostalCode,Country';

-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '<local directory path>' row format
delimited FIELDS TERMINATED BY '|' lines terminated by '\n' STORED AS
TEXTFILE
SELECT
coalesce(tmp2.record["AdditionalInputData"] ,''),
coalesce(tmp2.record["AddressBlock1"] ,''),
coalesce(tmp2.record["AddressBlock2"] ,''),
coalesce(tmp2.record["AddressLine1"] ,''),
coalesce(tmp2.record["AddressLine1.Input"] ,''),
coalesce(tmp2.record["ApartmentLabel"] ,''),
coalesce(tmp2.record["ApartmentNumber"] ,''),
coalesce(tmp2.record["AUS.Address.Class"] ,''),
coalesce(tmp2.record["AUS.Level.Number"] ,''),
coalesce(tmp2.record["AUS.Parcel.ID"] ,''),
coalesce(tmp2.record["AUS.PID"] ,''),
coalesce(tmp2.record["AUS.Principal.Pid"] ,''),
coalesce(tmp2.record["AUS.SA1"] ,''),
coalesce(tmp2.record["Building"] ,''),
coalesce(tmp2.record["CAN.Census.CD"] ,''),
coalesce(tmp2.record["CAN.Census.CMA"] ,''),
coalesce(tmp2.record["CAN.Census.CSD"] ,''),
coalesce(tmp2.record["CAN.Census.CT"] ,''),
coalesce(tmp2.record["CAN.Census.DA"] ,''),
coalesce(tmp2.record["CAN.FormattedStreet.Range"] ,''),
coalesce(tmp2.record["City"] ,''),
coalesce(tmp2.record["City.Input"] ,''),
coalesce(tmp2.record["City.Matched"] ,''),
coalesce(tmp2.record["CitySubdivision"] ,''),
coalesce(tmp2.record["CitySubdivision.Input"] ,''),
coalesce(tmp2.record["CitySubdivision.Matched"] ,''),
coalesce(tmp2.record["Confidence"] ,''),
coalesce(tmp2.record["CouldNotValidate"] ,''),
coalesce(tmp2.record["Country"] ,''),
coalesce(tmp2.record["Country.Input"] ,''),
coalesce(tmp2.record["County"] ,''),
coalesce(tmp2.record["County.Matched"] ,''),
coalesce(tmp2.record["FirmName"] ,''),
coalesce(tmp2.record["FirmName.Input"] ,''),
coalesce(tmp2.record["Firmname.Matched"] ,''),
coalesce(tmp2.record["GBR.Aliased.Locality"] ,''),
coalesce(tmp2.record["GBR.Dependent.Locality"] ,''),
coalesce(tmp2.record["GBR.DependentStreet.Name"] ,''),
coalesce(tmp2.record["GBR.DoubleDependent.Locality"] ,''),
coalesce(tmp2.record["GBR.Historic.Postcode"] ,''),
coalesce(tmp2.record["GBR.OSAPR"] ,''),
coalesce(tmp2.record["GBR.RPC"] ,''),
coalesce(tmp2.record["GBR.UPRN"] ,''),
```

```
coalesce(tmp2.record["HouseNumber"] ,''),
coalesce(tmp2.record["Housenumber.Matched"] ,''),
coalesce(tmp2.record["IRL.Eircode"] ,''),
coalesce(tmp2.record["ITA.Historical.Postcode"] ,''),
coalesce(tmp2.record["LeadingDirectional"] ,''),
coalesce(tmp2.record["MatchOnAllStreetFields"] ,''),
coalesce(tmp2.record["MatchOnStreetDirectional"] ,''),
coalesce(tmp2.record["MultimatchCount"] ,''),
coalesce(tmp2.record["NZL.Aliased.SUBURB"] ,''),
coalesce(tmp2.record["ParsedAddressLine1.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentLable.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentNumber.Input"] ,''),
coalesce(tmp2.record["ParsedCity.Input"] ,''),
coalesce(tmp2.record["ParsedCitySubDivision.Input"] ,''),
coalesce(tmp2.record["ParsedCountry.Input"] ,''),
coalesce(tmp2.record["ParsedHouseNumber.Input"] ,''),
coalesce(tmp2.record["ParsedPlaceName.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeAddOn.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeBase.Input"] ,''),
coalesce(tmp2.record["ParsedPostStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedPreStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvince.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["PlaceName"] ,''),
coalesce(tmp2.record["POBox"] ,''),
coalesce(tmp2.record["PostalCode"] ,''),
coalesce(tmp2.record["PostalCode.AddOn"] ,''),
coalesce(tmp2.record["PostalCode.Input"] ,''),
coalesce(tmp2.record["Postalcode.Matched"] ,''),
coalesce(tmp2.record["PrecisionCode"] ,''),
coalesce(tmp2.record["Principality"] ,''),
coalesce(tmp2.record["ProcessedBy"] ,''),
coalesce(tmp2.record["RecordID"] ,''),
coalesce(tmp2.record["StateProvince"] ,''),
coalesce(tmp2.record["StateProvince.Input"] ,''),
coalesce(tmp2.record["StateProvince.Matched"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Matched"] ,''),
coalesce(tmp2.record["Status"] ,''),
coalesce(tmp2.record["Status.Code"] ,''),
coalesce(tmp2.record["Status.Description"] ,''),
coalesce(tmp2.record["StreetName"] ,''),
coalesce(tmp2.record["StreetName.Matched"] ,''),
coalesce(tmp2.record["StreetType"] ,''),
coalesce(tmp2.record["StreetType.Matched"] ,''),
coalesce(tmp2.record["Subcity"] ,''),
coalesce(tmp2.record["TrailingDirectional"] ,''),
coalesce(tmp2.record["VendorCode"] ,'') FROM
(SELECT  addressvalidation(${hivevar:engineconf},
${hivevar:inputoption},${hivevar:header},
RecordID,AddressLine1,City,PostalCode,Country) as mygp from gavtable )
```

```
 as
 addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

*Reference data placed on HDFS and downloaded to working directory for jobs*

```
-- Register Global Addressing Module [GAV] BDQ Hive UDF Jar
ADD JAR <directory
path>/gam-globaladdressvalidation-hive-${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
 class names needed for this job to run.
CREATE TEMPORARY FUNCTION addressvalidation as
 'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';

-- set Engine configuration
set
hivevar:engineconf='{"dataDirPath":{"dataDir":"/home/hduser/gav/GGB062017.zip",
"referenceDataPathLocation":"HDFS"}, "productDatabaseInfoList":
[{"dbPath":"/home/hduser/gav/GGB062017.zip","countryCode":
["GBR"],"processType":"VALIDATE"}]}';

-- set input configuration
set hivevar:inputoption='{"casing":"Mixed","matchMode":"Relaxed",
"defaultCountry":"GBR","maximumResults":2,"returnInputAddress":false,
"returnParsedAddress":false,"returnPrecisionCode":false,"mustMatchAddressNumber":false,
"mustMatchStreet":false,"mustMatchCity":false,"mustMatchLocality":false,
"mustMatchState":false,"mustMatchStateProvince":false,
"mustMatchPostCode":false,"keepMultiMatch":true,"preferPostalOverCity":false,
"cityFallback":true,"postalFallback":true,"validationLevel":"ADDRESS"}';


-- set header
set
hivevar:header='inputkeyvalue,AddressLine1,AddressLine2,City,postalcode,
StateProvince,firmname,Country';

-- Execute Query on the desired table, to display the job output on
console.
 This query returns a map of key value pairs containing output fields
for each row.
SELECT * FROM (SELECT
addressvalidation(${hivevar:engineconf},${hivevar:inputoption},
${hivevar:header},inputkeyvalue,addressline1,AddressLine2,city,postalcode,
stateprovince,firmname,country)as mygp from address_validation) as
```

```
addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

# Universal Addressing Module Functions

## Using a Hive UDF of Universal Addressing Module

To run each Hive UDF job, you can either run these steps individually on your Hive client within a single session, or create an HQL file compiling all the required steps sequentially and run it in one go.

1. In your Hive client, log in to the required Hive database.
2. Register the JAR file of Spectrum™ Data & Address Quality for Big Data SDK UAM Module.

```
ADD JAR
/home/hduser/uam/uam.universaladdress.hive.${project.version}.jar;
```

3. Create an alias for the Hive UDF of the Address Quality job you want to run.

   **Note:** String in quotes represents the class names needed for this job to run.

   For example:

```
CREATE TEMPORARY FUNCTION addressvalidation as
'com.pb.bdq.uam.process.hive.addressvalidation.AddressValidationUDF';
```

4. Enable or disable the hive fetch task conversion.
   For Example:

```
set hive.fetch.task.conversion=none;
```

5. Use `hivevar:engineconf` to set the engine configurations. It includes details, such as database settings, COBOL runtime path, process type, DPV DB path, suiteLinkDBPath, ewsDBPath, rdiDBPath, lacsDBPath and preloading type
   For Example:

```
set
hivevar:engineconf='{"referenceData":{"dataDir":"/user/hduser/ReferenceData/
```

```
AddressQuality/UAM/Data.zip","referenceDataPathLocation":"HDFS"},"cobolRuntimePath":"",

"modulesDir":"","dpvDbPath":"/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip",

"suiteLinkDBPath":"/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip","ewsDBPath":

"/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip","rdiDBPath":null,"lacsDBPath":

"/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip"}';
```

6. Specify the settings for the input data using the `hivevar:inputoption` parameter.
   For example:

```
set
hivevar:inputoption='{"casing":"Mixed","matchMode":"Relaxed","defaultCountry":"GBR",

                        "maximumResults":2,"returnInputAddress":false,

"returnParsedAddress":false,"returnPrecisionCode":false,"returnMatchScore":true,

"mustMatchAddressNumber":false,"mustMatchStreet":false,"mustMatchCity":false,

"mustMatchLocality":false,"mustMatchState":false,"mustMatchStateProvince":false,

"mustMatchPostCode":false,"keepMultiMatch":true,"preferPostalOverCity":false,

                          "cityFallback":true,"postalFallback":true,
                          "validationLevel":"ADDRESS"}';
```

7. Set the general configurations, such as cacheSize, maxAddressObjectCount, and maxMemoryUsageMB, using `hivevar:generalconf`.
   For example:

```
set
hivevar:generalconf='{"cacheSize":"LARGE","maxThreadCount":8,"maxAddressObjectCount":8,
"rangesToExpand":"NONE","flexibleRangeExpansion":"ON","enableTransactionLogging":false,
"maxMemoryUsageMB":1024,"verbose":false}';
```

8. Specify the desired validation level to be used in a particular Hive job. Currently, only address validation is supported. So, set this value to VALIDATE.

For example;

```
set hivevar:processtype='VALIDATE';
```

9. Specify the header fields of the input table in comma-separated format, and assign to a variable or configuration property.

```
set
hivevar:header='inputkeyvalue,AddressLine1,AddressLine2,City,postalcode,
StateProvince,firmname,Country';
```

10. To run the job and display the job output on the console, write the query as indicated in this example:

    **Note:** This query returns a map of key value pairs containing output fields for each row.

```
SELECT
tmp2.record["HouseNumber"],tmp2.record["Confidence"],tmp2.record["AddressLine1"],
tmp2.record["StreetName"],tmp2.record["PostalCode"],tmp2.record["ElementInputStatus"],
tmp2.record["MailabilityScore"] FROM ( SELECT
globalvalidation(${hivevar:engineconf},
${hivevar:generalconf},${hivevar:inputconf},${hivevar:unlockCode},${hivevar:header},recordid,
addressline1,city,stateprovince,postalcode,country) as mygp from
address)
as addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record
 ;
```

To run the job and dump the job output in a designated file, write the query as indicated in this example:

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/GlobalAddressing/' row
 format delimited
FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS TEXTFILE
SELECT
tmp2.record["HouseNumber"],tmp2.record["Confidence"],tmp2.record["AddressLine1"],
tmp2.record["StreetName"],tmp2.record["PostalCode"],tmp2.record["ElementInputStatus"],
tmp2.record["MailabilityScore"] FROM ( SELECT
globalvalidation(${hivevar:engineconf},
${hivevar:generalconf},${hivevar:inputconf},${hivevar:unlockCode},${hivevar:header},
recordid,addressline1,city,stateprovince,postalcode,country) as mygp
 from address)
as addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record
 ;
```

**Note:** Use the alias defined earlier for the UDF.

# Validate Address

Validate Address standardizes and validates addresses using postal authority address data. It can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state or province names, and more.

> **Note:** Currently, Validate Address supports only US addresses.

Validate Address also returns result indicators about validation attempts, such as whether it validated the address, the level of confidence in the returned address, and the reason for failure if the address could not be validated.

During address matching and standardization, Validate Address separates address lines into components and compares those to the contents of the **Universal Addressing Module** databases. If a match is found, the input address is *standardized* to the database information. If no database match is found, Validate Address optionally *formats* the input addresses. The formatting process attempts to structure the address lines according to the conventions of the appropriate postal authority.

> **Note:** Validate Address supports CASS Certified™ processing which enables you to qualify for USPS® postal discounts.

**Attention:** Before creating and running the first Validate Address job, ensure the Acushare service is running. For steps, see **Running Acushare Service** on page 13.

## Sample Hive Script

### *Reference data placed on local node*

```
-- Register Universal Addressing Module - US [UAM-US] BDQ Hive UDAF Jar

ADD JAR <Directory
path>/uam.universaladdress.hive.${project.version}.jar;

-- Provide alias to UDAF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION uamvalidation as
'com.pb.bdq.uam.process.hive.universaladdress.UAMUSAddressingUDF';
-- set LD_LIBRARY_PATH(path to modules lib, runtime/lib and runtime/bin)

,G1RTS(path containing COBOL runtime) and ACU_RUNCBL_JNI_ONLOAD_DISABLE
 in this configuration
set
mapreduce.admin.user.env=LD_LIBRARY_PATH=/home/hduser/~/runtime/lib:/home/hduser/~/runtime/
bin:/home/hduser/~/server/modules/universaladdress/lib,ACU_RUNCBL_JNI_ONLOAD_DISABLE=1,G1RTS=/
```

```
home/hduser/~/ ;


-- set engine configuration
set
hivevar:engineconf='{"referenceData":{"dataDir":"/home/hduser/ReferenceData/AddressQuality/
UAM/Data","referenceDataPathLocation":"LocaltoDataNodes"},"cobolRuntimePath":"","modulesDir":"",
"dpvDbPath":"/home/hduser/ReferenceData/AddressQuality/UAM/Data","suiteLinkDBPath":"/home/hduser/
ReferenceData/AddressQuality/UAM/Data","ewsDBPath":"/home/hduser/ReferenceData/AddressQuality/UAM/
Data","rdiDBPath":"/home/hduser/ReferenceData/AddressQuality/UAM/Data","lacsDBPath":"/home/hduser/
ReferenceData/AddressQuality/UAM/Data"}';

-- set input configuration
set
hivevar:inputconf='{"processType":"VALIDATE","performUSProcessing":true,
"outputStandardAddress":true,"outputAddressElements":false,"outputPostalData":false,
"outputParsedInput":false,"outputAddressBlocks":false,"outputFormattedOnFail":false,
"outputCasing":"MIXED","outputPostalCodeSeparator":true,
"outputMultinationalCharacters":false,"performDPV":false,"performRDI":false,"performESM":false,
"performASM":false,"performEWS":false,\"performLACSLink":false,"performLOT":
false,"failOnCMRAMatch":false,"extractFirm":false,"extractUrb":false,"outputReport3553":false,
"outputReportSummary":true,"outputCASSDetail":false,
"outputFieldLevelReturnCodes":false,"keepMultimatch":false,"maximumResults":10,
"standardAddressFormat":"STANDARD_ADDRESS_FORMAT_COMBINED_UNIT",
"standardAddressPMBLine":"STANDARD_ADDRESS_PMB_LINE_NONE","cityNameFormat":"CITY_FORMAT_STANDARD",
"vanityCityFormatLong":true,"outputCountryFormat":"ENGLISH","homeCountry":"United
 States",
"streetMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM","firmMatchingStrictness":
"MATCHING_STRICTNESS_MEDIUM","directionalMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"dualAddressLogic":"DUAL_NORMAL","dpvSuccessfulStatusCondition":"DPV_CONDITON_ALWAYS",
"reportListFileName":"","reportlistProcessorName":"","reportlistNumber":1,"reportMailerAddress":
"","reportMailerName":"","reportMailerCityLine":"","addressLineSearchOnFail":true,
"outputStreetAlias":true,"outputVeriMoveBlock":false,"dpvDetermineNoStat":false,
"dpvDetermineVacancy":false,"outputAbbreviatedAlias":false,"outputPreferredAlias":false,
"outputPreferredCity":"CITY_OVERRIDE_NAME_ZIP4","performSuiteLink":false,
"suppressZplusPhantomCarrierR777":false,"dpvSeedList":null,"lacsSeedList":null,
"zipInputSet":null,"reportName":null,"jobRequest":false,"properties":{"DPVDetermineVacancy":"N",
"DualAddressLogic":"N","PerformASM":"N","ExtractUrb":"N","OutputCasing":"M",
"AddressLineSearchOnFail":"Y","ReportListFileName":"","ReportMailerCityLine":"",
"OutputFormattedOnFail":"N","OutputFieldLevelReturnCodes":"N","OutputStreetNameAlias":"Y",
"ReportListProcessorName":"","OutputAddressBlocks":"N","ExtractFirm":"N",
"DirectionalMatchingStrictness":"M","OutputPreferredCity":"Z","ReportListNumber":"1",
"FirmMatchingStrictness":"M","KeepMultimatch":"N","StandardAddressPMBLine":"N",
"OutputMultinationalCharacters":"N","PerformSuiteLink":"N","OutputShortCityName":"S",
"DPVSuccessfulStatusCondition":"A","PerformLACSLink":"N","PerformEWS":"N",
"OutputPostalCodeSeparator":"Y","FailOnCMRAMatch":"N","PerformLOT":"N",
"StandardAddressFormat":"C","SuppressZplusPhantomCarrierR777":"N","OutputCountryFormat":"E",
"OutputRecordType":"A    ","HomeCountry":"United
States","ReportMailerAddress":"",
"OutputReport3553":"N","OutputVeriMoveDataBlock":"N","PerformRDI":"N","ReportMailerName":"",
"OutputAbbreviatedAlias":"N","PerformESM":"N","PerformDPV":"N","OutputVanityCityFormatLong":"Y",
"OutputReportSummary":"Y","OutputPreferredAlias":"N","StreetMatchingStrictness":
"M","DPVDetermineNoStat":"N","MaximumResults":"10"}}';
```

```
-- set general configuration
set
hivevar:generalconf='{"dFileType":"SPLIT","dMemoryModel":"MEDIUM","lacsLinkMemoryModel":
"MEDIUM","suiteLinkMemoryModel":"MEDIUM"}';


-- set reference path
set
hivevar:location='/home/hduser/ReferenceData/AddressQuality/UAM/Data';

-- set process type
set hivevar:processtype='VALIDATE';

-- set header
set
hivevar:header='InputKeyValue,AddressLine1,AddressLine2,City,DefectNumber,FirmName,PostalCode,
StateProvince';

-- Execute Query on the desired table, to display the job output on
console. This query returns a
map of key value pairs containing output fields for each row.
SELECT tmp2.record["Confidence"],tmp2.record["AddressLine1"] FROM (
select uamvalidation
(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},${hivevar:inputconf},
${hivevar:header},inputkeyvalue,firmname,addressline1,addressline2,city,stateprovince,postalcode,
text) from uam_us) as addressgroup LATERAL VIEW
explode(addressgroup.mygp) tmp2 as record ;


-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/GlobalAddressing/' row
format delimited FIELDS
TERMINATED BY ',' lines terminated by '\n' STORED AS TEXTFILE
SELECT tmp2.record["Confidence"],tmp2.record["AddressLine1"] FROM (
select uamvalidation
(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},
${hivevar:inputconf},${hivevar:header},inputkeyvalue,firmname,addressline1,
addressline2,city,stateprovince,postalcode,text) from uam_us) as
addressgroup
LATERAL VIEW explode(addressgroup.mygp) tmp2 as record ;

+------------+--------------------+----------+---------------+-------------+------------+
| address.recordid  |     address.addressline1     |  address.city   |
 address.stateprovince  | address.postalcode  | address.country  |
+------------+--------------------+----------+---------------+-------------+------------+
| 1                 | 18 Merivale St               | South Brisbane  |
 QLD                    | 4101                | AUS              |
| 2                 | 19 Serpentine Rd             | Albany          |
 WA                     | 6330                | AUS              |
| 3                 | 317 VICTORIA ST GR           | BRUNSWICK       |
 VIC                    | 3056                | AUS              |
| 4                 | DUPLEX 6/16-18 O'CONNELL ST  | AINSLIE         |
```

```
 ACT                    | 2602                 | AUS              |
| 5                   | LOT 154 470 BRYGON CREEK DR  | UPPER COOMERA    |
 QLD                    | 4209                 | AUS              |
| 6                   | 16 GREENE ST                 | WARRAWONG        |
 ACT                    | 2502                 | AUS              |
| 7                   | UNIT 47/16 BLAIRMOUNT ST     | PARKINSON        |
 QLD                    | 4115                 | AUS              |
| 8                   | 13-15 FRANCESCO CRES         | BELLA VISTA      |
 NSW                    | 2153                 | AUS              |
| 9                   | 4 RYANS LANE                 | HEATHCOTE        |
 VIC                    | 3523                 | AUS              |
| 10                  | 1 CHRISTMAS LN               | NORTH POLE       |
 VIC                    | 1111                 | AUS              |
+------------+--------------------+----------+---------------+-------------+-----------+



+----------+-------------+-------+-----------------------------+-------------+
|Confidence |StreetName     |HouseNumber |     AddressLine1
  | AddressType   |
+----------+-------------+----------+---------------------------+-------------+
| 100.00   | MERIVALE      | 18    | 18 MERIVALE ST                | S
      |
| 99.42    | SERPENTINE    | 19       | 19 SERPENTINE RD E           |
 S       |
| 97.95    | VICTORIA      | 317      | 317 VICTORIA ST              |
 S       |
| 100.00   | O'CONNELL     | 16-18    | DUP 6 16-18 O'CONNELL ST     |
 S       |
| 0.00     | BRYGON CREEK  | 470      | LOT 154 470 BRYGON CREEK DR  |
 U       |
| 76.99    | GREENE        | 16        | 16 GREENE ST
| S       |
| 100.00   | BLAIRMOUNT    | 16       | U 47 16 BLAIRMOUNT ST        |
 S       |
| 100.00   | FRANCESCO     | 13-15    | 13-15 FRANCESCO CRES         |
 S       |
| 100.00   | RYANS         | 4        | 4 RYANS LANE                 |
 S       |
| 0.00     | CHRISTMAS     | 1        | 1 CHRISTMAS LN               |
 U       |
+----------+-------------+----------+---------------------------+-------------+
```

### *Reference data placed on HDFS and downloaded on local nodes for jobs*

```
-- Register Universal Addressing Module - US [UAM-US] BDQ Hive UDAF Jar

ADD JAR <Directory
path>/uam.universaladdress.hive.${project.version}.jar;

-- Provide alias to UDAF class (optional). String in quotes represent
class names needed for this
```

```
job to run.
CREATE TEMPORARY FUNCTION uamvalidation as
'com.pb.bdq.uam.process.hive.universaladdress.
UAMUSAddressingUDF';

set hive.fetch.task.conversion=none;

-- set LD_LIBRARY_PATH(path to modules lib, runtime/lib and runtime/bin)

,G1RTS(path containing COBOL runtime) and ACU_RUNCBL_JNI_ONLOAD_DISABLE

in this configuration
set
mapreduce.admin.user.env=LD_LIBRARY_PATH=/home/hduser/acushareInstall/modules/c1p/lib:/home/
hduser/acushareInstall/runtime/bin:/home/hduser/acushareInstall/runtime/lib,
ACU_RUNCBL_JNI_ONLOAD_DISABLE=1,G1RTS=/home/hduser/acushareInstall/runtime;


-- set engine configuration
set
hivevar:engineconf='{"referenceData":{"referenceDataPathLocation":"HDFS","dataDir":
"/user/hadoop/ReferenceData/UAM_US","dataDownloader":{"dataDownloader":"HDFS","localFSRepository":
"/opt/ReferenceData/UAM-US"}},"cobolRuntimePath":"","modulesDir":"","acushareServiceRunning":false,
"unixVersion":"REDHAT7","acushareLicensePath":"","dpvDbPath":"/user/hadoop/ReferenceData/UAM_US",
"suiteLinkDBPath":"/user/hadoop/ReferenceData/UAM_US","ewsDBPath":"/user/hadoop/ReferenceData/
UAM_US","rdiDBPath":"/user/hadoop/ReferenceData/UAM_US","lacsDBPath":"/user/hadoop/ReferenceData/
UAM_US"}';

-- set input configuration
set
hivevar:inputconf='{"processType":"VALIDATE","performUSProcessing":true,
"outputStandardAddress":true,"outputAddressElements":false,"outputPostalData":false,
"outputParsedInput":false,"outputAddressBlocks":false,"outputFormattedOnFail":false,
"outputCasing":"MIXED","outputPostalCodeSeparator":true,"outputMultinationalCharacters":false,
"performDPV":false,"performRDI":false,"performESM":false,"performASM":false,
"performEWS":false,"performLACSLink":false,"performLOT":false,"failOnCMRAMatch":false,
"extractFirm":false,"extractUrb":false,"outputReport3553":false,"outputReportSummary":true,
"outputCASSDetail":false,"outputFieldLevelReturnCodes":false,"keepMultimatch":false,
"maximumResults":10,"standardAddressFormat":"STANDARD_ADDRESS_FORMAT_COMBINED_UNIT",
"standardAddressPMBLine":"STANDARD_ADDRESS_PMB_LINE_NONE","cityNameFormat":
"CITY_FORMAT_STANDARD","vanityCityFormatLong":true,"outputCountryFormat":
"ENGLISH","homeCountry":"United States","streetMatchingStrictness":
"MATCHING_STRICTNESS_MEDIUM","firmMatchingStrictness"
:"MATCHING_STRICTNESS_MEDIUM","directionalMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"dualAddressLogic":"DUAL_NORMAL","dpvSuccessfulStatusCondition":
"DPV_CONDITON_ALWAYS","reportListFileName":"","reportlistProcessorName":"",
"reportlistNumber":1,"reportMailerAddress":"","reportMailerName":"",
"reportMailerCityLine":"","addressLineSearchOnFail":true,"outputStreetAlias":true,
"outputVeriMoveBlock":false,"dpvDetermineNoStat":false,"dpvDetermineVacancy":false,
"outputAbbreviatedAlias":false,"outputPreferredAlias":false,"outputPreferredCity":
"CITY_OVERRIDE_NAME_ZIP4","performSuiteLink":false,"suppressZplusPhantomCarrierR777":
false,"dpvSeedList":null,"lacsSeedList":null,"zipInputSet":null,
"reportName":null,"jobRequest":false,"properties":{"DPVDetermineVacancy":"N",
```

```
"DualAddressLogic":"N","PerformASM":"N","ExtractUrb":"N","OutputCasing":"M",
"AddressLineSearchOnFail":"Y","ReportListFileName":"",
"ReportMailerCityLine":"","OutputFormattedOnFail":"N","OutputFieldLevelReturnCodes":"N",
"OutputStreetNameAlias":"Y","ReportListProcessorName":"","OutputAddressBlocks":"N","ExtractFirm":
"N","DirectionalMatchingStrictness":"M","OutputPreferredCity":"Z","ReportListNumber":"1",
"FirmMatchingStrictness":"M","KeepMultimatch":"N","StandardAddressPMBLine":"N",
"OutputMultinationalCharacters":"N","PerformSuiteLink":"N","OutputShortCityName":"S",
"DPVSuccessfulStatusCondition":"A","PerformLACSLink":"N","PerformEWS":"N",
"OutputPostalCodeSeparator":"Y","FailOnCMRAMatch":"N","PerformLOT":
"N","StandardAddressFormat":"C","SuppressZplusPhantomCarrierR777":"N",
"OutputCountryFormat":"E","OutputRecordType":"A ",
"HomeCountry":"United
States","ReportMailerAddress":"","OutputReport3553":"N",
"OutputVeriMoveDataBlock":"N","PerformRDI":"N","ReportMailerName":"","OutputAbbreviatedAlias":
"N","PerformESM":"N","PerformDPV":"N","OutputVanityCityFormatLong":"Y","OutputReportSummary":"Y",
"OutputPreferredAlias":"N","StreetMatchingStrictness":"M","DPVDetermineNoStat":"N","MaximumResults":
"10"}}';

-- set general configuration
set
hivevar:generalconf='{"dFileType":"SPLIT","dMemoryModel":"MEDIUM","lacsLinkMemoryModel":
"MEDIUM","suiteLinkMemoryModel":"MEDIUM"}';

-- set process type
set hivevar:processtype='VALIDATE';

-- set header
set
hivevar:header='InputKeyValue,AddressLine1,AddressLine2,City,DefectNumber,FirmName,PostalCode,
StateProvince';

-- Execute Query on the desired table, to display the job output on
console. This query returns a
map of key value pairs containing output fields for each row.
SELECT
tmp2.record["Status"],tmp2.record["Status.Description"],tmp2.record["Confidence"],
tmp2.record["AddressLine1"],tmp2.record["InputKeyValue"] FROM ( select
 uamvalidation
(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},${hivevar:inputconf},
${hivevar:header},inputkeyvalue,addressline1,addressline2,city,defectnumber,firmname,
postalcode,stateprovince) as mygp from address_uam) as addressgroup
LATERAL VIEW explode
(addressgroup.mygp) tmp2 as record ;
```

### *Reference data placed on HDFS and downloaded to working directory for jobs*

```
-- Register Universal Addressing Module - US [UAM-US] BDQ Hive UDAF Jar

ADD JAR <Directory
path>/uam.universaladdress.hive.${project.version}.jar;

-- Provide alias to UDAF class (optional). String in quotes represent
```

```
class names needed for this
job to run.
CREATE TEMPORARY FUNCTION uamvalidation as
'com.pb.bdq.uam.process.hive.universaladdress.
UAMUSAddressingUDF';

--Provide reference data zip file to be added to cache
ADD ARCHIVE
hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/Data.zip;

set hive.fetch.task.conversion=none;

-- set LD_LIBRARY_PATH(path to modules lib, runtime/lib and runtime/bin)
 ,G1RTS(path containing
COBOL runtime) and ACU_RUNCBL_JNI_ONLOAD_DISABLE in this configuration
set
mapreduce.admin.user.env=LD_LIBRARY_PATH=/home/hduser/acushareInstall/modules/c1p/lib:/home/
hduser/acushareInstall/runtime/bin:/home/hduser/acushareInstall/runtime/lib,
ACU_RUNCBL_JNI_ONLOAD_DISABLE=1,G1RTS=/home/hduser/acushareInstall/runtime;

-- set engine configuration
set
hivevar:engineconf='{"referenceData":{"dataDir":"/user/hduser/ReferenceData/AddressQuality/
UAM/Data.zip","referenceDataPathLocation":"HDFS"},"cobolRuntimePath":"","modulesDir":"","dpvDbPath":
"/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip","suiteLinkDBPath":"/user/hduser/
ReferenceData/AddressQuality/UAM/Data.zip","ewsDBPath":"/user/hduser/ReferenceData/AddressQuality/
UAM/Data.zip","rdiDBPath":null,"lacsDBPath":"/user/hduser/ReferenceData/AddressQuality/
UAM/Data.zip"}';

-- set input configuration
set
hivevar:inputconf='{"processType":"VALIDATE","performUSProcessing":true,
"outputStandardAddress":true,"outputAddressElements":false,"outputPostalData"
:false,"outputParsedInput":false,"outputAddressBlocks":false,"outputFormattedOnFail":false,
"outputCasing":"MIXED","outputPostalCodeSeparator":true,"outputMultinationalCharacters":false,
"performDPV":false,"performRDI":false,"performESM":false,"performASM":false,
"performEWS":false,"performLACSLink":false,"performLOT":false,"failOnCMRAMatch":false,
"extractFirm":false,"extractUrb":false,"outputReport3553":false,"outputReportSummary":true,
"outputCASSDetail":false,"outputFieldLevelReturnCodes":false,"keepMultimatch":false,
"maximumResults":10,"standardAddressFormat":"STANDARD_ADDRESS_FORMAT_COMBINED_UNIT",
"standardAddressPMBLine":"STANDARD_ADDRESS_PMB_LINE_NONE","cityNameFormat":"CITY_FORMAT_STANDARD",
"vanityCityFormatLong":true,"outputCountryFormat":"ENGLISH","homeCountry":"United
 States",
"streetMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM","firmMatchingStrictness"
:"MATCHING_STRICTNESS_MEDIUM","directionalMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"dualAddressLogic":"DUAL_NORMAL","dpvSuccessfulStatusCondition":"DPV_CONDITON_ALWAYS",
"reportListFileName":"","reportlistProcessorName":"","reportlistNumber":1,"reportMailerAddress":"",
"reportMailerName":"","reportMailerCityLine":"","addressLineSearchOnFail"
:true,"outputStreetAlias":true,"outputVeriMoveBlock":false,"dpvDetermineNoStat":false,
"dpvDetermineVacancy":false,"outputAbbreviatedAlias":false,"outputPreferredAlias":false,
"outputPreferredCity":"CITY_OVERRIDE_NAME_ZIP4","performSuiteLink":false,
"suppressZplusPhantomCarrierR777":false,"dpvSeedList":null,"lacsSeedList":null,"zipInputSet":null,
"reportName":null,"jobRequest":false,"properties":{"DPVDetermineVacancy":"N","DualAddressLogic":"N",
```

```
"PerformASM":"N","ExtractUrb":"N","OutputCasing":"M","AddressLineSearchOnFail":"Y",
"ReportListFileName":"","ReportMailerCityLine":
"","OutputFormattedOnFail":"N","OutputFieldLevelReturnCodes":"N","OutputStreetNameAlias":"Y",
"ReportListProcessorName":"","OutputAddressBlocks":"N","ExtractFirm":"N",
"DirectionalMatchingStrictness":"M","OutputPreferredCity":"Z","ReportListNumber":"1",
"FirmMatchingStrictness":"M","KeepMultimatch":"N","StandardAddressPMBLine":"N",
"OutputMultinationalCharacters":"N","PerformSuiteLink":"N","OutputShortCityName":"S",
"DPVSuccessfulStatusCondition":"A","PerformLACSLink":"N","PerformEWS":"N",
"OutputPostalCodeSeparator":"Y","FailOnCMRAMatch":"N","PerformLOT":"N","StandardAddressFormat":
"C","SuppressZplusPhantomCarrierR777":"N","OutputCountryFormat":"E","OutputRecordType":"A

","HomeCountry":"United
States","ReportMailerAddress":"","OutputReport3553":"N",
"OutputVeriMoveDataBlock":"N","PerformRDI":"N","ReportMailerName":"","OutputAbbreviatedAlias":
"N","PerformESM":"N","PerformDPV":"N","OutputVanityCityFormatLong":"Y","OutputReportSummary":
"Y","OutputPreferredAlias":"N","StreetMatchingStrictness":"M","DPVDetermineNoStat":"N",
"MaximumResults":"10"}}';

-- set general configuration
set
hivevar:generalconf='{"dFileType":"SPLIT","dMemoryModel":"MEDIUM","lacsLinkMemoryModel":
"MEDIUM","suiteLinkMemoryModel":"MEDIUM"}';

-- set reference path
set
hivevar:location='/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip';

-- set process type
set hivevar:processtype='VALIDATE';

-- set header
set hivevar:header='InputKeyValue,AddressLine1,AddressLine2,
City,DefectNumber,FirmName,PostalCode,StateProvince';

-- Execute Query on the desired table, to display the job output on
console.
 This query returns a map of key value pairs containing output fields
for each row.
SELECT
tmp2.record["Status"],tmp2.record["Status.Description"],tmp2.record["Confidence"],
tmp2.record["AddressLine1"],tmp2.record["InputKeyValue"] FROM ( select
 uamvalidation
(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},${hivevar:inputconf},
${hivevar:header},inputkeyvalue,addressline1,addressline2,city,defectnumber,firmname,postalcode,
stateprovince) as mygp from address_uam) as addressgroup LATERAL VIEW
explode(addressgroup.mygp)
 tmp2 as record ;
```

# Validate Address Global

Validate Address Global provides enhanced address standardization and validation for addresses outside the U.S. and Canada. Validate Address Global can also validate addresses in the U.S. and Canada but its strength is validation of addresses in other countries. If you process a significant number of addresses outside the U.S. and Canada, you should consider using Validate Address Global.

Validate Address Global is part of the Universal Addressing Module.

Validate Address Global performs several steps to achieve a quality address, including parsing, validation, and formatting.

## *Address Parsing, Formatting, and Standardization*

Restructuring incorrectly fielded address data is a complex and difficult task especially when done for international addresses. People introduce many ambiguities as they enter address data into computer systems. Among the problems are misplaced elements (such as company or personal names in street address fields) or varying abbreviations that are not only language, but also country specific. Validate Address Global identifies address elements in address lines and assigns them to the proper fields. This is an important precursor to the actual validation. Without restructuring, "no match" situations might result.

Properly identified address elements are also important when addresses have to be truncated or shortened to fit specific field length requirements. With the proper information in the right fields, specific truncation rules can be applied.

- Parses and analyzes address lines and identifies individual address elements
- Processes over 30 different character sets
- Formats addresses according to the postal rules of the country of destination
- Standardizes address elements (such as changing AVENUE to AVE)

## *Global Address Validation*

Address validation is the correction process where properly parsed address data is compared against reference databases supplied by postal organizations or other data providers. Validate Address Global validates individual address elements to check for correctness using sophisticated fuzzy matching technology and produces standardized and formatted output based on postal standards and user preferences. FastCompletion validation type can be used in quick address entry applications. It allows input of truncated data in several address fields and generates suggestions based on this input.

In some cases, it is not possible to fully validate an address. Here Validate Address Global has a unique deliverability assessment feature that classifies addresses according to their probable deliverability.

## Sample Hive Scripts

### *Reference data placed on local node*

```
-- Register Universal Addressing Module [UAM-Global] BDQ Hive UDAF Jar

ADD JAR <Directory path>/uam.global.hive.${project.version}.jar;

ADD FILE <Directory path>/libAddressDoctor5.so;

-- Provide alias to UDAF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION globalvalidation as
'com.pb.bdq.uam.process.hive.global.GlobalAddressingUDF';

-- set Engine configuration
set
hivevar:engineconf='[{"referenceData":{"dataDir":"/home/hduser/ReferenceData/AddressQuality/
Global","referenceDataPathLocation":"LocaltoDataNodes"},"databaseType":"BATCH_INTERACTIVE","preloadingType":"NONE",
"allCountries":true,"supportedCountries":"CAN,USA,AUS"}]';

-- set input configuration
set
hivevar:inputconf='{"resultStateProvinceType":"COUNTRY_STANDARD","processMatchingScope":
"ALL","inputForceCountryISO3":null,"inputDefaultCountryISO3":"USA","inputFormatDelimiter":"CRLF",
"resultFormatDelimiter":"CRLF","resultIncludeInputs":false,"resultCountryType":"NAME_EN",
"processOptimizationLevel":"STANDARD","resultPreferredLanguage":"DATABASE","processMode":"BATCH",
"resultPreferredScript":"DATABASE","resultMaximumResults":1,"resultCasing":"NATIVE","properties":
{"Result.MaximumResults":"1","Database.AddressGlobal":"Database","Input.FormatDelimiter":"CRLF",
"Process.Mode":"BATCH","Input.ForceCountryISO3":"","Result.CountryType":"NAME_EN",
"Process.OptimizationLevel":"STANDARD","Result.IncludeInputs":"false","Input.DefaultCountryISO3"
:"USA","Process.EnrichmentAMAS":"false","Result.PreferredScript":"DATABASE","Process.MatchingScope"
:"ALL","Result.Casing":"NATIVE","Result.PreferredLanguage":"DATABASE","Result.StateProvinceType"
:"COUNTRY_STANDARD","Result.FormatDelimiter":"CRLF"}}';

-- set general configuration
set hivevar:generalconf='{"cacheSize":"LARGE","maxThreadCount":8,
"maxAddressObjectCount":8,"rangesToExpand":"NONE","flexibleRangeExpansion":"ON",
"enableTransactionLogging":false,"maxMemoryUsageMB":1024,"verbose":false}';

-- set unlock codec
set hivevar:unlockCode='';

-- set header
set
hivevar:header='recordid,AddressLine1,City,StateProvince,PostalCode,Country';

-- Execute Query on the desired table, to display the job output on
console.
 This query returns a map of key value pairs containing output fields
for each row.
SELECT tmp2.record["HouseNumber"],tmp2.record["Confidence"],
```

```
tmp2.record["AddressLine1"],tmp2.record["StreetName"],tmp2.record["PostalCode"],
tmp2.record["ElementInputStatus"],tmp2.record["MailabilityScore"] FROM

( SELECT
globalvalidation(${hivevar:engineconf},${hivevar:generalconf},${hivevar:inputconf},${hivevar:unlockCode},
${hivevar:header},recordid,addressline1,city,stateprovince,postalcode,country)

 as mygp from address) as addressgroup LATERAL VIEW
explode(addressgroup.mygp) tmp2 as record ;


-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/GlobalAddressing/'
row format delimited FIELDS TERMINATED BY ',' lines terminated by '\n'
 STORED AS TEXTFILE
SELECT tmp2.record["HouseNumber"],tmp2.record["Confidence"],
tmp2.record["AddressLine1"],tmp2.record["StreetName"],tmp2.record["PostalCode"],
tmp2.record["ElementInputStatus"],tmp2.record["MailabilityScore"] FROM

( SELECT  globalvalidation(${hivevar:engineconf},${hivevar:generalconf},
${hivevar:inputconf},${hivevar:unlockCode},${hivevar:header},recordid,addressline1,
city,stateprovince,postalcode,country) as mygp from address) as
addressgroup
 LATERAL VIEW explode(addressgroup.mygp) tmp2 as record ;

+------------+--------------------+----------+---------------+-------------+------------+
| address.recordid  |     address.addressline1     |  address.city   |
 address.stateprovince  | address.postalcode  | address.country  |
+------------+--------------------+----------+---------------+-------------+------------+
| 1                 | 18 Merivale St               | South Brisbane  |
 QLD                    | 4101                | AUS              |
| 2                 | 19 Serpentine Rd             | Albany          |
 WA                     | 6330                | AUS              |
| 3                 | 317 VICTORIA ST GR           | BRUNSWICK       |
 VIC                    | 3056                | AUS              |
| 4                 | DUPLEX 6/16-18 O'CONNELL ST  | AINSLIE         |
 ACT                    | 2602                | AUS              |
| 5                 | LOT 154 470 BRYGON CREEK DR  | UPPER COOMERA   |
 QLD                    | 4209                | AUS              |
| 6                 | 16 GREENE ST                 | WARRAWONG       |
 ACT                    | 2502                | AUS              |
| 7                 | UNIT 47/16 BLAIRMOUNT ST     | PARKINSON       |
 QLD                    | 4115                | AUS              |
| 8                 | 13-15 FRANCESCO CRES         | BELLA VISTA     |
 NSW                    | 2153                | AUS              |
| 9                 | 4 RYANS LANE                 | HEATHCOTE       |
 VIC                    | 3523                | AUS              |
| 10                | 1 CHRISTMAS LN               | NORTH POLE      |
 VIC                    | 1111                | AUS              |
+------------+--------------------+----------+---------------+-------------+------------+
```

```
+----------+-------------+-------+-----------------------------+-------------+
|Confidence |StreetName     |HouseNumber |     AddressLine1
  | AddressType   |
+----------+-------------+-------+-----------------------------+-------------+
| 100.00   | MERIVALE      | 18    | 18 MERIVALE ST                | S
      |
| 99.42    | SERPENTINE    | 19       | 19 SERPENTINE RD E           |
 S       |
| 97.95    | VICTORIA      | 317      | 317 VICTORIA ST              |
 S       |
| 100.00   | O'CONNELL     | 16-18    | DUP 6 16-18 O'CONNELL ST     |
 S       |
| 0.00     | BRYGON CREEK  | 470      | LOT 154 470 BRYGON CREEK DR  |
 U       |
| 76.99    | GREENE        | 16        | 16 GREENE ST
| S       |
| 100.00   | BLAIRMOUNT    | 16       | U 47 16 BLAIRMOUNT ST        |
 S       |
| 100.00   | FRANCESCO     | 13-15    | 13-15 FRANCESCO CRES         |
 S       |
| 100.00   | RYANS         | 4        | 4 RYANS LANE                 |
 S       |
| 0.00     | CHRISTMAS     | 1        | 1 CHRISTMAS LN               |
 U       |
+----------+-------------+-------+-----------------------------+-------------+
```

### *Reference data placed on HDFS and downloaded on local nodes for jobs*

```
-- Register Universal Addressing Module [UAM-Global] BDQ Hive UDAF Jar

ADD JAR <Directory path>/uam.global.hive.${project.version}.jar;

ADD FILE <Directory path>/libAddressDoctor5.so;

-- Provide alias to UDAF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION globalvalidation as
'com.pb.bdq.uam.process.hive.global.GlobalAddressingUDF';

-- set Engine configuration
set hivevar:engineconf='[{"referenceData":{"referenceDataPathLocation"
:"HDFS","dataDir":"/user/hadoop/ReferenceData/Global","dataDownloader":{"dataDownloader":
"HDFS","localFSRepository":"/opt/PitneyBowes/UAM_Global"}},"databaseType":"BATCH_INTERACTIVE",
"preloadingType":"PARTIAL","allCountries":true,"supportedCountries":"ALL"}]';



-- set input configuration
set hivevar:inputconf='{"resultStateProvinceType":"COUNTRY_STANDARD",
"processMatchingScope":"ALL","inputForceCountryISO3":null,"inputDefaultCountryISO3":"USA",
"inputFormatDelimiter":"CRLF","resultFormatDelimiter":"CRLF","resultIncludeInputs":false,
```

```
"resultCountryType":"NAME_EN","processOptimizationLevel":"STANDARD","resultPreferredLanguage":
"DATABASE","processMode":"BATCH","resultPreferredScript":"DATABASE","resultMaximumResults":1,
"resultCasing":"NATIVE","properties":{"Result.MaximumResults":"1","Database.AddressGlobal":"Database",
"Input.FormatDelimiter":"CRLF","Process.Mode":"BATCH","Input.ForceCountryISO3":"","Result.CountryType":
"NAME_EN","Process.OptimizationLevel":"STANDARD","Result.IncludeInputs":"false",
"Input.DefaultCountryISO3":"USA","Process.EnrichmentAMAS":"false","Result.PreferredScript":
"DATABASE","Process.MatchingScope":"ALL","Result.Casing":"NATIVE","Result.PreferredLanguage"
:"DATABASE","Result.StateProvinceType":"COUNTRY_STANDARD","Result.FormatDelimiter":"CRLF"}}';


-- set general configuration
set
hivevar:generalconf='{"cacheSize":"LARGE","maxThreadCount":8,"maxAddressObjectCount":8,
"rangesToExpand":"NONE","flexibleRangeExpansion":"ON","enableTransactionLogging":false,
"maxMemoryUsageMB":1024,"verbose":false}';

-- set unlock codec
set hivevar:unlockCode='';

-- set header
set hivevar:header='InputKeyValue,AddressLine1,AddressLine2,
City,PostalCode,StateProvince,FirmName,Country';

-- Execute Query on the desired table, to display the job output on
console.
This query returns a map of key value pairs containing output fields
for each row.
SELECT
tmp2.record["Country"],tmp2.record["Confidence"],tmp2.record["MailabilityScore"],
tmp2.record["HouseNumber"],tmp2.record["AddressLine1"],tmp2.record["StreetName"],
tmp2.record["PostalCode"],tmp2.record["ElementInputStatus"] FROM
(SELECT  globalvalidation(${hivevar:engineconf},${hivevar:generalconf},
${hivevar:inputconf},${hivevar:unlockCode},${hivevar:header},inputkeyvalue,
addressline1,addressline2,city,postalcode,stateprovince,firmname,country)

 as mygp from address_global) as addressgroup LATERAL VIEW explode
(addressgroup.mygp) tmp2 as record ;
```

*Reference data placed on HDFS and downloaded to working directory for jobs*

```
-- Register Universal Addressing Module [UAM-Global] BDQ Hive UDAF Jar

ADD JAR <Directory path>/uam.global.hive.${project.version}.jar;

ADD FILE <Directory path>/libAddressDoctor5.so;

-- Provide alias to UDAF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION globalvalidation as
'com.pb.bdq.uam.process.hive.global.GlobalAddressingUDF';

-- Add Reference Data zipped Files
```

```
ADD ARCHIVE
hdfs:///user/hduser/ReferenceData/AddressQuality/AD/AD2032017_590.zip;
ADD ARCHIVE
hdfs:///user/hduser/ReferenceData/AddressQuality/AD/AD3032017_590.zip;

-- set Engine configuration
set
hivevar:engineconf='[{"referenceData":{"dataDir":"/user/hduser/ReferenceData/AddressQuality/AD/AD2032017_590.zip",
"referenceDataPathLocation":"HDFS"},"databaseType":"BATCH_INTERACTIVE",
"preloadingType":"NONE","allCountries":false,"supportedCountries":"ALL"},
{"referenceData":{"dataDir":"/user/hduser/ReferenceData/AddressQuality/AD/AD3032017_590.zip",
"referenceDataPathLocation":"HDFS"},"databaseType":"BATCH_INTERACTIVE",
"preloadingType":"NONE","allCountries":false,"supportedCountries":"ALL"}]';


-- set input configuration
set hivevar:inputconf='{"resultStateProvinceType":"COUNTRY_STANDARD",
"processMatchingScope":"ALL","inputForceCountryISO3":null,"inputDefaultCountryISO3":
"USA","inputFormatDelimiter":"CRLF","resultFormatDelimiter":"CRLF","resultIncludeInputs":false,
"resultCountryType":"NAME_EN","processOptimizationLevel":"STANDARD","resultPreferredLanguage"
:"DATABASE","processMode":"BATCH","resultPreferredScript":"DATABASE","resultMaximumResults":1,
"resultCasing":"NATIVE","properties":{"Result.MaximumResults":"1","Database.AddressGlobal":
"Database","Input.FormatDelimiter":"CRLF","Process.Mode":"BATCH","Input.ForceCountryISO3":
"","Result.CountryType":"NAME_EN","Process.OptimizationLevel":"STANDARD","Result.IncludeInputs"
:"false","Input.DefaultCountryISO3":"USA","Process.EnrichmentAMAS":"false",
"Result.PreferredScript":"DATABASE","Process.MatchingScope":"ALL","Result.Casing":"NATIVE",
"Result.PreferredLanguage":"DATABASE","Result.StateProvinceType":"COUNTRY_STANDARD",
"Result.FormatDelimiter":"CRLF"}}';

-- set general configuration
set
hivevar:generalconf='{"cacheSize":"LARGE","maxThreadCount":8,"maxAddressObjectCount":8,
"rangesToExpand":"NONE","flexibleRangeExpansion":"ON","enableTransactionLogging":false,
"maxMemoryUsageMB":1024,"verbose":false}';
-- set unlock codec
set hivevar:unlockCode='';

-- set header
set
hivevar:header='InputKeyValue,AddressLine1,AddressLine2,City,PostalCode,StateProvince,FirmName,Country';

-- Execute Query on the desired table, to display the job output on
console.
 This query returns a map of key value pairs containing output fields
for each row.
SELECT
tmp2.record["Country"],tmp2.record["Confidence"],tmp2.record["MailabilityScore"],
tmp2.record["HouseNumber"],tmp2.record["AddressLine1"],tmp2.record["StreetName"],tmp2.record
["PostalCode"],tmp2.record["ElementInputStatus"]
FROM (SELECT
globalvalidation(${hivevar:engineconf},${hivevar:generalconf},
${hivevar:inputconf},${hivevar:unlockCode},${hivevar:header},inputkeyvalue,
addressline1,addressline2,city,postalcode,stateprovince,firmname,country)
```

```
as mygp from address_global) as addressgroup LATERAL VIEW
explode(addressgroup.mygp) tmp2 as record ;
```

## Validate Address Loqate

Validate Address Loqate standardizes and validates addresses using postal authority address data. Validate Address Loqate can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state/province names.

Validate Address Loqate also returns result indicators about validation attempts, such as whether or not Validate Address Loqate validated the address, the level of confidence in the returned address, the reason for failure if the address could not be validated, and more.

During address matching and standardization, Validate Address Loqate separates address lines into components and compares them to the contents of the Universal Addressing Module databases. If a match is found, the input address is standardized to the database information. If no database match is found, ValidateAddress Loqate optionally formats the input addresses. The formatting process attempts to structure the address lines according to the conventions of the appropriate postal authority.Validate Address Loqate is part of the Universal Addressing Module.

### Sample Hive Script

```
-- Register Universal Address Module [UAM] BDQ Hive Loqate UDAF Jar
ADD JAR <Directory path>/uam.loqate.hive.${project.version}.jar;

-- Provide alias to UDAF class (optional).
String in quotes represent class names needed for this job to run.
CREATE TEMPORARY FUNCTION loqatevalidation as
'com.pb.bdq.uam.process.hive.loqate.LoqateAddressingUDF';

-- Adding required files to distributed cache.
ADD FILES <Directory Path>/loqate-core.car;
ADD FILES <Directory Path>/LoqateVerificationLevel.csv;
ADD FILES <Directory Path>/Loqate.csv;
ADD FILES <Directory Path>/countryTables.csv;
ADD FILES <Directory Path>/countryNameTables.csv;

set hive.map.aggr = false;

-- set process configuration
set set
hivevar:processconf='{"processType":"VALIDATE","includeMatchedAddressElements":
false,"includeStandardizedInputAddressElements":false,"returnAddressDataBlocks":false,
"casing":"Mixed","outputReportSummary":false,"includeResultCodesforIndividualFields":false,
"returnMultipleAddresses":false,"failedOnMultiMatchFound":false,"countryFormat":"ENGLISH",
```

```
"defaultCountry":"USA","scriptAlphabet":"InputScript","returnGeocodedAddressFields":false,
"acceptanceLevel":"Level0","minimumMatchScore":0,"formatDataUsingAMASConventions":false,
"singleFieldDuplicateHandling":false,"multiFieldDuplicateHandling":false,
"nonStandardFieldDuplicateHandling":false,"outputFieldDuplicateHandling":false,
"returnMultipleAddressCount":10,"duplicateHandling":false,"includeStandardAddress":true}';

-- set general configuration
set
hivevar:generalconf='{"maxIdle":null,"minIdle":16,"maxActive":16,"maxWait":null,
"whenExhaustedAction":null,"testOnBorrow":null,"testOnReturn":null,"testWhileIdle":null,
"timeBetweenEvictionRunsMillis":null,"numTestsPerEvictionRun":null,"minEvictableIdleTimeMillis":null}';


-- set engine configuration
set
hivevar:engineconf='{"verbose":true,"toolInfo":true,"outputAddressFormat":false,
"logInput":false,"logOutput":false,"logFileName":null,"matchScoreAbsoluteThreshold":60,
"matchScoreThresholdFactor":95,"postalCodeMaxResults":10,"strictReferenceMatch":false}';

-- set reference directory path
set hivevar:location='/home/hduser/ReferenceData/AddressQuality/Loqate';

-- set process type
set hivevar:processtype='VALIDATE';


-- set input header
set
hivevar:header='InputKeyValue,AddressLine1,City,StateProvince,PostalCode,Country';


select SELECT
tmp2.record["HouseNumber"],tmp2.record["Confidence"],tmp2.record["AddressLine1"],
tmp2.record["StreetName"],tmp2.record["PostalCode"],tmp2.record["DPID"],tmp2.record["Barcode"]

 FROM ( SELECT
loqatevalidation(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},
${hivevar:processconf},${hivevar:location},${hivevar:header},inputkeyvalue,addressline1,city,stateprovince,
postalcode,country) as mygp from address) as <TABLE_NAME> LATERAL VIEW
 explode(addressgroup.mygp) tmp2 as record ;

-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/loqate/' row format
delimited
FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS TEXTFILE
SELECT * FROM
( SELECT
tmp2.record["HouseNumber"],tmp2.record["Confidence"],tmp2.record["AddressLine1"]
,tmp2.record["StreetName"],tmp2.record["PostalCode"],tmp2.record["DPID"],tmp2.record["Barcode"]

 FROM ( SELECT
loqatevalidation(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},
${hivevar:processconf},${hivevar:location},${hivevar:header},inputkeyvalue,addressline1,city,
```

```
stateprovince,postalcode,country) as mygp from address) as <TABLE_NAME>

 LATERAL VIEW explode(addressgroup.mygp) tmp2 as record ;

--Sample Input
+-------------+-----------------------------+-------------+----------+-------------+
| inputkeyvalue  |            addressline1            | stateprovince
 | postalcode  |    country    |
+-------------+-----------------------------+-------------+----------+-------------+
| 1              | 80 Quan Su                          |
 |             | Vietnam       |
| 2              | Final Av. Panteón Foro Libertador  |
 | 1010        | Venezuela     |
| 3              | P O Box 834                         |
 |             | St Vincent    |
| 4              | Colonia 2066                        |
 |             | Uruguay       |
| 5              | Ave de la Resistance BP127          |
 |             | Burkina Faso  |
| 6              | Buyuk Turon Street, 41              |
 |             | Uzbekistan    |
| 7              | Empire State Building               | NY
 | 10118       | US            |
| 8              | 3 Leontovycha St                    |
 |             | Ukraine       |
| 9              |                                     | Ceredigion
 |             | Wales         |
| 10             | 5 Main Street                       | Ballindalloch
 |             | Scotland      |
+-------------+-----------------------------+-------------+----------+-------------+


-- Sample Output
+----------+--------------+--------+---+-----------------------------------+
|Match Score|StreetName     |HouseNumber |            addressline1
        |
+----------+--------------+-----------+-----------------------------------+
| 100.00   | MERIVALE      | 80    | 80 Quan Su
  |
| 100.00    | SERPENTINE    |        | Final Av. Panteón Foro Libertador
  |
| 0.00    | VICTORIA      | 0       | P O Box 834
   |
| 75.00   | O'CONNELL     | 2066    | Colonia 2066
   |
| 83.33     | BRYGON CREEK  | 470      | Ave de la Resistance BP127
      |
| 100.00    | GREENE        |         | Buyuk Turon Street, 41
     |
| 96.8254   | BLAIRMOUNT    | 41       | Empire State Building
      |
| 83.950   | FRANCESCO     | 350     | 3 Leontovycha St
    |
```

```
| 50.00    | RYANS          | 3          |
    |
| 100      | CHRISTMAS      | 5          | 5 Main Street
    |
+----------+----------------+------------+-----------------------------------+

!quit
```

# Universal Name Module Functions

## Using a Hive UDF of Universal Name Module

To run each Hive UDF job, you can either run these steps individually on your Hive client within a single session, or create an HQL file compiling all the required steps sequentially and run it in one go.

1. In your Hive client, log in to the required Hive database.
2. Register the JAR file of Spectrum™ Data & Address Quality for Big Data SDK UNM Module.

   ```
   ADD JAR <Directory path>/unm.hive.${project.version}.jar;
   ```

3. Create an alias for the Hive UDF of the Data Quality job you wish to run.
   For example:

   ```
   CREATE TEMPORARY FUNCTION opennameparser as
   'com.pb.bdq.unm.process.hive.
   opennameparser.OpenNameParserUDF';
   ```

4. Specify the reference data path.
   - **Reference data is on HDFS**
     - Reference data is to be **downloaded to a working directory** for jobs
       - If the reference data is in *unarchived* file format, set the reference directory as:

         ```
         set
         hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
         "dataDir":"./referenceData","dataDownloader":{"dataDownloader":"DC"}}';
         ```

- If the reference data is in *archived* format, set the reference directory as:

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"./referenceData.zip","dataDownloader":
{"dataDownloader":"DC"}}';
```

- Reference data is to be **downloaded on local nodes** for jobs. In this case, set the reference data directory as:

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"/home/data/dm/referenceData","dataDownloader":{"dataDownloader":
"HDFS","localFSRepository":"/local/download"}}';
```

- **Reference data is on local path**: Ensure that data is present on each node of the cluster on the same path.

  Set the reference directory as:

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"LocaltoDataNodes",
"dataDir":"/home/data/referenceData"}';
```

5. Specify the configurations and other details for the job, and assign these to respective variables or configuration properties.

   **Note:** The rule must be in JSON format.

   For example,

```
set hivevar:rule='{"name":"name", "culture":"",
"splitConjoinedNames":false, "shortcutThreshold":0,
"parseNaturalOrderPersonalNames":false,
"naturalOrderPersonalNamesPriority":1,
"parseReverseOrderPersonalNames":false,
"reverseOrderPersonalNamesPriority":2,
"parseConjoinedNames":false,
"naturalOrderConjoinedPersonalNamesPriority":3,
"reverseOrderConjoinedPersonalNamesPriority":4,
"parseBusinessNames":false, "businessNamesPriority":5}';
```

   **Note:** Use the configuration properties in the respective job configurations. For example, `pb.bdq.match.rule`, `pb.bdq.match.express.column`, and `pb.bdq.consolidation.sort.field` where indicated in the respective sample HQL files.

6. Specify the header fields of the input table in comma-separated format, and assign to a variable or configuration property.

```
set hivevar:header='inputrecordid,Name,nametype';
```

7. To run the job and display the job output on the console, write the query as indicated in this example:

```
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"] from (select
opennameparser(${hivevar:rule}, ${hivevar:refdir}, ${hivevar:header},
inputrecordid, name, nametype) as tmp1 from nameparser) as tmp LATERAL
 VIEW explode(tmp1) adTable AS adid;
```

   To run the job and dump the job output in a designated file, write the query as indicated in the below example:

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/opennameparser/' row
format delimited FIELDS TERMINATED BY ',' lines terminated by '\n'
STORED AS TEXTFILE
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"] from (select
opennameparser(${hivevar:rule}, ${hivevar:refdir},
${hivevar:header}, inputrecordid, name, nametype) as tmp1 from
nameparser) as tmp LATERAL VIEW explode(tmp1) adTable AS adid;
```

   **Note:** Use the alias defined earlier for the UDF.

## Open Name Parser

OpenNameParser breaks down personal and business names and other terms in the name data field into their component parts. These parsed name elements are then subsequently available to other automated operations such as name matching, name standardization, or multi-record name consolidation.

OpenNameParser does the following:

- Determines the type of a name in order to describe the function that the name performs. Name entity types are divided into two major groups: personal names and business names. Within each of these major groups are subgroups.
- Determines the form of a name in order to understand which syntax the parser should follow for parsing. Personal names usually take on a natural (signature) order or a reverse order. Business names are usually ordered hierarchically.
- Determines and labels the component parts of a name so that the syntactical relationship of each name part to the entire name is identified. The personal name syntax includes prefixes, first,

middle, and last name parts, suffixes, and account description terms, among other personal name parts. The business name syntax includes the firm name and suffix terms.

- Parses conjoined personal and business names and either retains them as one record or splits them into multiple records. Examples of conjoined names include "Mr. and Mrs. John Smith" and "Baltimore Gas & Electric dba Constellation Energy".
- Parses output as records or as a list.
- Assigns a parsing score that reflects the degree of confidence that the parsing is correct.

## Sample Hive Script

### *Reference data placed on local node*

```
-- Register Universal Name Module [UNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/unm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional).
 String in quotes represent class names needed for this job to run.
-- Open Name Parser is implemented as a UDF (User Defined function).
Hence it processes one row at a time and generates a map of key value
pairs for each row.
CREATE TEMPORARY FUNCTION opennameparser as
'com.pb.bdq.unm.process.hive.opennameparser.OpenNameParserUDF';

-- set rule
set hivevar:rule='{"name":"name", "culture":"",
"splitConjoinedNames":false,
"shortcutThreshold":0, "parseNaturalOrderPersonalNames":false,
"naturalOrderPersonalNamesPriority":1,"parseReverseOrderPersonalNames":false,
"reverseOrderPersonalNamesPriority":2, "parseConjoinedNames":false,
"naturalOrderConjoinedPersonalNamesPriority":3,
"reverseOrderConjoinedPersonalNamesPriority":4,
"parseBusinessNames":false, "businessNamesPriority":5}';

-- Set Reference Directory. This must be a local path on cluster machines
 and must
be present on each node of the cluster at the same path.
set hivevar:refereceDataDetails='{"referenceDataPathLocation":
"LocaltoDataNodes","dataDir":"/home/data/referenceData"}';

-- set header
set hivevar:header='inputrecordid,Name,nametype';

set hive.fetch.task.conversion=none;
-- Execute Query on the desired table, to display the job output on
console.
 This query returns a map of key value pairs containing output fields
for each row.
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"]
from (select opennameparser(${hivevar:rule},
```

```
${hivevar:refereceDataDetails}, ${hivevar:header},
 inputrecordid, name, nametype) as tmp1 from nameparser) as
tmp LATERAL VIEW explode(tmp1) adTable AS adid;


-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/opennameparser/' row
format
delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS
TEXTFILE
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"]
 from (select opennameparser(${hivevar:rule},
${hivevar:refereceDataDetails}, ${hivevar:header},
 inputrecordid, name, nametype) as tmp1 from nameparser)
 as tmp LATERAL VIEW explode(tmp1) adTable AS adid;


--sample input data
+-------------------------------+------------------------+---------------------+
| inputrecordid        | name          |      nametype          |
+-------------------------------+------------------------+---------------------+
| 1                                |JOHN VAN DER LINDEN-JONES | Simple
 Name            |
| 2                                |RYAN JOHN SMITH           | Simple
 Name     |
+-------------------------------+------------------------+---------------------+

--sample output data
+----------------------------+-------------+---------------------+
| Name           | NameScore    |      CultureCode    |
+----------------------------+-------------+---------------------+
| JOHN VAN DER LINDEN-JONES  |   75         |      True           |
| RYAN JOHN SMITH            |   100        |      True           |
+----------------------------+-----+-------------------------------+
```

*Reference data placed on HDFS and downloaded on local nodes for jobs*

```
-- Register Universal Name Module [UNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/unm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in
quotes represent class names needed for this job to run.
-- Open Name Parser is implemented as a UDF (User Defined function).
Hence it processes one row at a time and generates a map of key value
pairs for each row.
CREATE TEMPORARY FUNCTION opennameparser as
'com.pb.bdq.unm.process.hive.opennameparser.OpenNameParserUDF';

-- set rule
set hivevar:rule='{"name":"name", "culture":"",
"splitConjoinedNames":false,
```

```
 "shortcutThreshold":0, "parseNaturalOrderPersonalNames":false,
"naturalOrderPersonalNamesPriority":1,
"parseReverseOrderPersonalNames":false,
 "reverseOrderPersonalNamesPriority":2, "parseConjoinedNames":false,
"naturalOrderConjoinedPersonalNamesPriority":3,
"reverseOrderConjoinedPersonalNamesPriority":4,
 "parseBusinessNames":false, "businessNamesPriority":5}';

-- Set Reference Directory. This must be a local path on cluster machines
 and
 must be present on each node of the cluster at the same path.
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS","dataDir"
:"/home/data/dn/referenceData","dataDownloader":{"dataDownloader":"HDFS","localFSRepository":"/local/download"}}';

-- set header
set hivevar:header='inputrecordid,Name,nametype';

set hive.fetch.task.conversion=none;
-- Execute Query on the desired table, to display the job output
 on console. This query returns a map of key value pairs containing
output fields for each row.
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"]
from (select opennameparser(${hivevar:rule},
${hivevar:refereceDataDetails},
 ${hivevar:header}, inputrecordid, name, nametype) as tmp1 from
nameparser)
as tmp LATERAL VIEW explode(tmp1) adTable AS adid;


-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/opennameparser/' row
format
delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS
TEXTFILE
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"]
 from (select opennameparser(${hivevar:rule},
${hivevar:refereceDataDetails},
 ${hivevar:header},inputrecordid, name, nametype) as tmp1 from
nameparser)
as tmp LATERAL VIEW explode(tmp1) adTable AS adid;

--sample input data
+-------------------------------+-----------------------+----------------------+
| inputrecordid        | name                       |      nametype
    |
+-------------------------------+-----------------------+----------------------+
| 1                                  |JOHN VAN DER LINDEN-JONES |
Simple Name       |
| 2                                  |RYAN JOHN SMITH           |
Simple Name       |
```

```
+------------------------------+----------------------+---------------------+

--sample output data
+----------------------------+--------------+---------------------+
| Name           | NameScore    |      CultureCode    |
+----------------------------+--------------+---------------------+
| JOHN VAN DER LINDEN-JONES  |   75         |       True          |
| RYAN JOHN SMITH            |   100        |       True          |
+----------------------------+-----+-------------------------------+
```

*Reference data placed on HDFS and downloaded to working directory for jobs*

```
-- Register Universal Name Module [UNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/unm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in
quotes represent class names needed for this job to run.
-- Open Name Parser is implemented as a UDF (User Defined function).
Hence it processes one row at a time and generates a map of key value
pairs for each row.
CREATE TEMPORARY FUNCTION opennameparser as
'com.pb.bdq.unm.process.hive.opennameparser.OpenNameParserUDF';

-- HDFS Reference data- via Distributed Cache DC- unarchive
ADD FILES hdfs://<HOST>:<PORT>/home/hduser/referenceData/;

--HDFS reference data- via Distributed cache -archive form of reference
 data
--ADD ARCHIVE hdfs://<HOST>:<PORT>/home/hduser/referenceData.zip;

-- set rule
set hivevar:rule='{"name":"name", "culture":"",
"splitConjoinedNames":false,
 "shortcutThreshold":0, "parseNaturalOrderPersonalNames":false,

"naturalOrderPersonalNamesPriority":1,"parseReverseOrderPersonalNames":false,

 "reverseOrderPersonalNamesPriority":2, "parseConjoinedNames":false,
 "naturalOrderConjoinedPersonalNamesPriority":3,
"reverseOrderConjoinedPersonalNamesPriority":4,
 "parseBusinessNames":false, "businessNamesPriority":5}';

-- Set Reference Data details .

--reference data details, can be added in zip or unarchive form, dataDir
 symbolises reference data
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS","dataDir":"./referenceData",
"dataDownloader":{"dataDownloader":"DC"}}';

-- below format for archive form
--set
```

```
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS","dataDir":
"./referenceData.zip","dataDownloader":{"dataDownloader":"DC"}}';

-- set header
set hivevar:header='inputrecordid,Name,nametype';

set hive.fetch.task.conversion=none;
-- Execute Query on the desired table, to display the job
 output on console. This query returns a map of key value
 pairs containing output fields for each row.
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"]
from (select opennameparser(${hivevar:rule},
${hivevar:refereceDataDetails},
 ${hivevar:header}, inputrecordid, name, nametype) as tmp1 from
nameparser)
 as tmp LATERAL VIEW explode(tmp1) adTable AS adid;


-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/opennameparser/'
row format delimited FIELDS TERMINATED BY ','
 lines terminated by '\n' STORED AS TEXTFILE
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"]
from (select opennameparser(${hivevar:rule},
${hivevar:refereceDataDetails},
${hivevar:header}, inputrecordid, name, nametype) as tmp1 from
nameparser)
as tmp LATERAL VIEW explode(tmp1) adTable AS adid;



--sample input data
+-------------------------------+-----------------------+----------------------+
| inputrecordid        | name                       |      nametype
    |
+-------------------------------+-----------------------+----------------------+
| 1                                |JOHN VAN DER LINDEN-JONES |
Simple Name       |
| 2                                |RYAN JOHN SMITH           |
Simple Name       |
+-------------------------------+-----------------------+----------------------+

--sample output data
+----------------------------+--------------+----------------------+
| Name           | NameScore    |      CultureCode    |
+----------------------------+--------------+----------------------+
| JOHN VAN DER LINDEN-JONES  |   75         |       True    |
| RYAN JOHN SMITH            |   100        |       True          |
+----------------------------+-----+--------------------------------+
```

# 7 - Reporting Counters

## In this section

# Reporting Counters

The reporting counters are displayed only when you run a MapReduce or Spark job. The output is displayed either on the console or is dumped as a file, based on the query syntax you use. For hive jobs, reporting counters are displayed on Resource Manager of the hadoop cluster.

# Advanced Matching Module

## Interflow Match Report

Interflow Match locates matches between similar data records across two input record streams. The first record stream is a source for suspect records and the second stream is a source for candidate records.

Using match group criteria (for example a match key), Interflow Match identifies a group of records that are potentially duplicates of a particular suspect record.

*Reporting*

The Interflow Match job allows you to monitor the results of the job. The counters available are:

| | |
|---|---|
| **DUPLICATE_COLLECTIONS** | The number of duplicate collections, which consist of a suspect and its duplicate records grouped together by a CollectionNumber. |
| **EXPRESS_MATCHES** | The number of Express Matches made in a collection.<br><br>An Express Match is made when a suspect and candidate have an exact match on the contents of a designated field, usually an ExpressMatchKey provided by the Match Key Generator. If an Express Match is made, no further processing is done to determine if the suspect and candidate are duplicates. |
| **AVERAGE_SCORE** | The average match score of all duplicates.<br><br>The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match. |
| **INPUT_SUSPECTS** | The number of records in the input stream that the matcher tried to match to other records. |

| | |
|---|---|
| **SUSPECTS_WITH_DUPLICATES** | The number of input suspects that matched at least one candidate record. |
| **UNIQUE_SUSPECTS** | The number of input suspects that did not match any candidate records. |
| **SUSPECTS_WITH_CANDIDATES** | The number of input suspects that had at least one candidate record in its match group and therefore had at least one match attempt. |
| **SUSPECTS_WITHOUT_CANDIDATES** | The number of input suspects that had no candidate records in its match group and therefore had no match attempts. |
| **TOTAL_DUPLICATE_CANDIDATES** | The total number of duplicate candidates found. |
| **TOTAL_DUPLICATE_SCORE** | The total match score of all the duplicates. |

## Intraflow Match Report

Intraflow Match locates matches between similar data records within a single input stream. You can create hierarchical rules based on any fields that have been defined or created in other stages of the dataflow.

### *Reporting*

The Intraflow Match job allows you to monitor the results of the job. The counters available are:

| | |
|---|---|
| **INPUT_RECORDS** | The number of records in the matching stage before the matching sort is performed. |
| **DUPLICATE_RECORDS** | The number of duplicate records within a match group, which can be either a suspect or a candidate record. |
| **UNIQUE_RECORDS** | The number of suspect or candidate records which do not match any other records in their respective match group.<br><br>If it is the only record in a match group, a suspect is automatically unique. |
| **MATCH_GROUPS** | (Group By) Records grouped together by a match key. |
| **DUPLICATE_COLLECTIONS** | The number of duplicate collections, which consist of a suspect and its duplicate records grouped together by a CollectionNumber. |
| **EXPRESS_MATCHES** | The number of Express Matches made in a collection.<br><br>An Express Match is made when a suspect and candidate have an exact match on the contents of a designated field, usually an ExpressMatchKey provided by the Match Key Generator. If an Express Match is made, no further processing is done to determine if the suspect and candidate are duplicates. |

| | |
|---|---|
| **AVERAGE_SCORE** | The average match score of all duplicates.<br>The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match. |
| **TOTAL_DUPLICATES** | The total number of duplicates found. |
| **TOTAL_SCORE** | The total match score of all duplicates. |

## Transactional Match Report

Transactional Match matches suspect records against candidate records of a group of records to identify duplicates. The records are first grouped by a selected column, post which the first record is marked as the suspect record. All the remaining records of the group, termed as candidate records, are matched against the suspect record.

If the candidate record is a duplicate, it is assigned a collection number, the match record type is labeled a Duplicate, and the record is then written out. Any unmatched candidates in the group are assigned a collection number of 0, labeled as Unique and then written out as well.

### *Reporting*

The Transactional Match job allows you to monitor the results of the job. The counters available are:

| | |
|---|---|
| **AVERAGE_SCORE** | The average match score of all duplicates.<br>The possible values are 0-100, with 0 indicating a poor match and 100 indicating an exact match. |
| **INPUT_SUSPECTS** | The number of records in the input stream that the matcher tried to match to other records. |
| **SUSPECTS_WITH_DUPLICATES** | The number of input suspects that matched at least one candidate record. |
| **UNIQUE_SUSPECTS** | The number of input suspects that did not match any candidate records. |
| **SUSPECTS_WITH_CANDIDATES** | The number of input suspects that had at least one candidate record in its match group and therefore had at least one match attempt. |
| **SUSPECTS_WITHOUT_CANDIDATES** | The number of input suspects that had no candidate records in its match group and therefore had no match attempts. |
| **TOTAL_DUPLICATES_SCORE** | The total match score of all duplicates. |
| **TOTAL_DUPLICATES** | The total number of duplicates found. |

# Global Addressing Module

## Global Address Validation Report

The counters provide the reporting statistics across all supported countries for which the Global Address Validation job is run.

For a list of supported countries, refer to **ISO Country Codes and Coder Support** on page 369. The counters available are:

### *Counters of Records*

- *TOTAL_INPUT*- The total number of input records.
- *TOTAL_COUNTRY_RECORDS*- The total number of input records processed for the country listed.

### *Counters of Matched Elements*

This section provides summary information about matched elements for each country.

- *HOUSE_NUMBER_MATCHED*- The number of records that matched on the house number.
- *STREET_NAME_MATCHED*- The number of records that matched on the street name.
- *CTY_MATCHED*- The number of records that matched on the city name.
- *POST_CODE_MATCHED*- The number of records that matched on the postal code.
- *STATE_PROVINCE_MATCHED*- The number of records that matched on the state/province.

### *Counters of Confidence Levels*

This section provides statistics about the number of records for each country that matched at different confidence levels. The confidence level assigned to a returned address ranges from zero (0) to 100. *Zero* indicates failure. *100* indicates a very high level of confidence that the match results are correct.

- *CONF_LOW*- The records that match at a confidence level less than 40, such as *CONF_10*, and *CONF_20*.
- *CONF_MID*- The records that match at a confidence level between 40 and 85, such as *CONF_50*, and *CONF_80*.
- *CONF_HIGH*- The records that match at a confidence level greater than 85, such as *CONF_90*, and *CONF_100*.
- *Confidence*- The total number of records whose confidence level is checked. This value is same as the total number of records.

### *Precision Code Counters*

This section provides statistics on the number of records in your job that matched for each precision code. The precision code describes the level of precision for each record's address match.

### *Precision Code Z Category*

The Z category indicates that a match was made at the postal code level.

- *PRECISION_Z1*- The number of records that match to the ZIP Code or postal code 1.
- *PRECISION_Z2*- The number of records that result in a ZIP + 2 or partial match to postal code 2.
- *PRECISION_Z3*- The number of records that match to ZIP + 4 or postal code 2.

### *Precision Code G Category*

Geographic level candidates return a precision code beginning with the letter G. The number following the G in the precision code provides more detailed information about the accuracy of the match.

- *PRECISION_G1*- The number of records that match to state/province.
- *PRECISION_G2*- The number of records that match to county/region.
- *PRECISION_G3*- The number of records that match to city/town.
- *PRECISION_G4*- The number of records that match to suburb/village.

### *Precision Code S Category*

Street level candidates return a precision code beginning with the letter S. The character following the S in the precision code provides more detailed information about the accuracy of the match.

- *PRECISION_SX*- The number of records that validated at a street intersection.
- *PRECISION_SC*- The number of records that matched at the house-level projected from the nearest segment.
- *PRECISION_SL*- The number of records that matched to a sublocality (block or sector) at street level.
- *PRECISION_SG*- The number of records that matched to a center of a locality (areaName3) or Locality level geocode derived from topographic feature.
- *PRECISION_S0*- The number of records where parts of the address may have matched the source data.
- *PRECISION_S1*- The number of records that matched to a point located at a ZIP Code.
- *PRECISION_S2*- The number of records that matched to a point located at ZIP + 2.
- *PRECISION_S3*- The number of records that matched to a point located at a ZIP + 4.
- *PRECISION_S4*- The number of records that matched at the street level.
- *PRECISION_S5*- The number of records that matched to the street address.
- *PRECISION_S6*- The number of records that matched to a point located at point ZIP centroid.
- *PRECISION_S7*- The number of records that matched to a street address that was interpolated between houses.
- *PRECISION_S8*- The number of records that matched to the street address or house number.

*Precision Code B Category*

- *PRECISION_B1*- The number of records that matched to an unvalidated PO Box.
- *PRECISION_B2*- The number of records that matched to a validated PO Box.

# Universal Addressing Module

## Validate Address Reports

For Validate Address you can generate these three types of reports:

**Note:** For instructions on how to generate reports, see the *Dataflow Designer Guide*.

1. Address Validation Summary Report
2. CASS Report 3553 Summary Report
3. CASS Detail Report

**Note:** CASS Report 3553 and CASS Detail Report can be generated in CASS Certified™ mode.

### Address Validation Summary Report

The Address Validation Summary displays these details:

- **Statistics about the job**- Number of input records, number of address recods processed, total number of records for which address validation was attempted, total number of records matched successfully, total number of unmatched records, and number of standard addresses returned successfully.
- **Statistics about validation/correction of records**- Number of original postal codes confirmed through address match, number of postal codes corrected through address match, number of original postal codes retained, number of postal codes not available, and total number of records for which address validation was attempted.
- **Statistics about matched records**- Total number of records which were valid on input, total number of records corrected, and total number of records successfully matched.
- **Statistics about unmatched records**- Number of records having street mismatch, number of records having house mismatch, number of records having number range mismatch and total number of unmatched records.
- **Statistics about records processed**- Matched and unmatched records processed for US, Canada and International. The total number of records processed are also displayed.

**Note:** Additionally, Validate Address Summary also shows a graphical representation of matching summary, matched records, unmatched records, and total records.

## CASS 3553 Summary Report

The USPS CASS 3553 report must be submitted to the USPS along with the mailing to qualify for certain discounts. The report displays these details in a tabular form:

**Software**

- Name of the CASS certified company and software, along with the version and configuration of the software.
- Name of the Z4Change certified company and the software, along with the version and configuration of the software.
- Name of the DirectDPV certified company and the software, along with the version and configuration of the software.

**List**

- Name of the company that coded the address list and/or performed ZIP+4 matching.
- Processing date for each list.
- The date of each database package used.
- The name or identification number of the address list.
- The number of lists used to produce the mailing.
- The total number of address records submitted when the list was coded.

**Output**- Summarized output of the CASS process. Consists of total number of records verified at these levels:

- Zip+4
- DPV
- Z4 Change
- Direct DPV
- 5 Digit carrier route
- Carrier Route (CR-RT)
- enhanced Line Of Travel (eLOT) sequence number

Validation period of the verified records are also displayed.

**Mailer**- Signature, name and address of the individual who processed the list and the date on which the form was signed.

**Qualitative Statistical Summary (QSS)**- Information which allows list processors to evaluate the quality of their address list before its contents enter the mailstream.

- High Rise Default- The number of records which were successfully matched at the street address level but could not be matched to a valid secondary address.
- High Rise Exact- The number of records which were exactly matched to a primary and secondary address.

- Rural RTE Default- The number of rural route address records which were verified at the route number level but could not be matched to a valid primary range.
- Rural RTE Exact- The number of rural route address records which were matched at the route number level and could be matched to a valid primary range.
- Locatable Address Conversion System (LACS)/ LACSLink System- The number of address records which have been converted through LACS/LACSLink System.
- EWS (Early Warning System)- The new address records which are not present in the current US Postal Service's ZIP + 4 File.
- SuiteLink- The number of addresses which are corrected by SuiteLink.

For details about USPS Form 3553, see **www.usps.com**.

### CASS Detail Report

CASS Certified™ processing also generates the USPS CASS Detail Report. This report is not mandatory to be submitted for postal discounts. The report displays these details:

**Job Identification and General Information**- Date of processing and name of the job.

**CASS Run Information**

- Software Name
- Total number of records processed
- Total number of records Zip Coded
- Total number of records failed to Zip Code
- ZIP+4 Match Statistics such as, PO Box, RR Default, RR Exact, Street, and more.
- Processing Information such as number of input address matched and not matched to the ZIP+4.

**DPV Run Information**

**LACS and LACSLink Run Information**

**SuiteLink Run Information**

> **Note:** The matched, unmatched results and validity of DPV, LACS and LACSLink, and Suitelink are explained as footnotes at the end of the CASS detail report.

For more information about the CASS settings while using the SDK, see **Using a Validate Address MapReduce Job** on page 134 and **Using a Validate Address Spark Job** on page 136.

## Validate Address Global Report

The counters provide the reporting statistics across all supported countries for which the Validate Address Global job is run.

For a list of supported countries, refer to **ISO Country Codes and Coder Support** on page 369. The counters available are:

### *Summary Counters*

The summary counters provide a summation of the values of each particular counter type across countries.

For example, the counter `SUMMARY_FAILED_COUNT` is the sum of the values of the `FAILED_COUNT` counter for all the supported countries in which a particular Validate Address Global job is run.

- Status details: Lists the validations and correction results for each country.
  - SUMMARY_STATUS_I4_COUNT— Addresses that could not be corrected but that are very likely to be deliverable.
  - SUMMARY_STATUS_I2_COUNT— Addresses that could not be corrected and are unlikely to be deliverable.
  - SUMMARY_STATUS_V_COUNT— Addresses that were correct on input.
  - SUMMARY_STATUS_C_COUNT— Addresses that were corrected by Validate Address Global.
  - SUMMARY_STATUS_I3_COUNT— Addresses that could not be corrected but have a fair chance that the address is deliverable.
  - SUMMARY_STATUS_S_COUNT— Addresses that were successfully parsed.
  - SUMMARY_FAILED_COUNT— Addresses that could not be verified, corrected, or parsed.
  - COUNTRY— A comma-separated list of the country codes for which the address validation is run.
  - SUMMARY_CASING— The casing method of the output. For details, refer to the *Options* section of the *Validate Address Global* stage in the *Addressing Guide*.
- Job summary: Lists summary of the job
  - SUMMARY_END_TIME— The date and time that the job ended.
  - SUMMARY_START_TIME— The date and time that the job started.
  - SUMMARY_CHARSET— Character sets processed.
  - SUMMARY_DEFAULT_COUNTRY— The default country specified in the Default country (ISO3 format) option.

### *Counters specific to countries*

These counters provide the reporting statistics for the various supported countries. Each counter label begins with the country code to which the counter value corresponds.

For example, these counters provide the reporting statistics for United States:

1. UNITEDSTATES_STATUS_I4_COUNT
2. UNITEDSTATES_STATUS_S_COUNT
3. UNITEDSTATES_STATUS_I3_COUNT
4. UNITEDSTATES_FAILED_COUNT
5. UNITEDSTATES_STATUS_I2_COUNT
6. UNITEDSTATES_STATUS_C_COUNT
7. UNITEDSTATES_STATUS_V_COUNT

Similar counters are listed for all the supported countries for which you have run the Validate Address Global job.

## Validate Address Loqate Report

The Validate Address Loqate job displays these report counters:

### *Input Name/Address*

- **Input Record Count**—The total number of input addresses for the job.
- **Address Records Processed**—The total number of input addresses for the job.
- **Total Records For Which Address Validation Attempted**—The number of input records for which validation was attempted.
- **Total Records Successfully Matched**—The number of input addresses that were validated or corrected. This is the number of input addresses that did not result in a status of "F".
- **Total Unmatched Records**—The number of input addresses that could not be validated or corrected. This is equal to the number of input addresses that resulted in a status of "F".
- **Standard Address Returned Successfully**—The number of unmatched (failed) addresses that Validate Global Address standardized. Standardization only happens if the option **Return standardized data when no match is found** is enabled. For more information, see **Output Data Options**.

### *Postal code validation and correction*

- **Original Postal Code Confirmed Via Address Match**—The number of addresses whose ACR component status for the postal code is 2.
- **Postal Code Corrected Via Address Match**—The number of input postal codes that were incorrect but were corrected by Validate Global Address.
- **Original Postal Code Retained**—The number of addresses whose ACR component status for the postal code is 1.
- **No Postal Code Available**—The postal data contained no postal code for the address..

### *Input addresses that matched to known addresses in the database*

1. **Total Records Valid On Input**—The number of addresses that were confirmed to be correct.
2. **Total Corrected**—The number of addresses that Validate Global Address corrected.
3. **Total Records Successfully Matched**—The total number of addresses that were either validated or corrected successfully.

### *Input addresses that could not be confirmed or corrected*

1. **Street Mismatch**—The number of addresses whose street could not be validated or corrected.
2. **House Mismatch**—The number of addresses whose house number that could not be validated or corrected.

3. **Total Unmatched Records**—The total number of addresses that could not be validated or corrected.

*Records processed*

- **RecordsProcessedByLOQATE**—The total number of records processed in the job.

# Universal Name Module

## Open Name Parser Report

The Open Name Parser provides summary statistics about the job, such as the total number of input records and the total number of records that contained no name data, as well as several parsing statistics.

*General Results*

| | |
|---|---|
| **INPUT_RECORDS** | The number of records in the input. |
| **NO_NAME_DATA_RECORDS** | The number of records in the input that do not contain name data to be parsed. |
| **NAMES_PARSED_OUT** | The number of names in the input which were parsed. |
| **LOWEST_NAME_PARSING_SCORE** | The lowest parsing score given to any name in the input. |
| **HIGHEST_NAME_PARSING_SCORE** | The highest parsing score given to any name in the input. |
| **AVERAGE_NAME_PARSING_SCORE** | The average parsing score given among all parsed names in the input. |

*Personal Name Parsing Results*

| | |
|---|---|
| **PERSONAL_NAME_RECORDS** | The number of personal names in the input. |
| **CONJOINED_NAMES_PARSED** | The number of parsed names from records that contained conjoined names.<br><br>For example, if your input had five records with two conjoined names, and seven records with three conjoined names, this counter value for this field is 31, according to the equation: (5 x 2) + (7 x 3). |
| **TWO_CONJOINED_NAMES_RECORDS** | The number of input records containing two conjoined names. |

| | |
|---|---|
| **THREE_CONJOINED_NAMES_RECORDS** | The number of input records containing three conjoined names. |
| **TITLE_OF_RESPECT_NAMES** | The number of parsed names containing a title of respect. |
| **MATURITY_SUFFIX_NAMES** | The number of parsed names containing a maturity suffix. |
| **GENERAL_SUFFIX_NAMES** | The number of parsed names containing a general suffix. |
| **ACCOUNT_DESCRIPTION_PERSONAL_NAMES** | The number of parsed names containing an account description. |
| **TOTAL_REVERSE_ORDER_NAMES** | The number of parsed names in the reverse order, resulting in the output field `IsReverseOrder` as "True". |

*Business Name Parsing Results*

| | |
|---|---|
| **BUSINESS_NAME_RECORDS** | The number of input records containing business names. |
| **FIRM_SUFFIX_NAMES** | The number of parsed names containing a firm suffix. |
| **ACCOUNT_DESCRIPTION_BUSINESS_NAMES** | The number of input records containing an account description. |
| **TOTAL_DBA_RECORDS** | The number of input records containing Doing Business As (DBA) conjunctions, resulting in both output fields `isPersonal` and `isFirm` as "True". |
| **TOTAL_PARSED** | The total number of names parsed. |
| **TOTAL_NAME_PARSING_SCORE** | The total parsing score of all names. |

# Appendix

## In this section

# A - Exceptions

## In this section

# Exception Messages

### *Exceptions - Java API*

- `<Classname>.<Member>` is null or empty.
- `GroupbyMROption.numReduceTasks = 0` min values should be 1.
- `maxNumOfDuplicates` = 0 min values should be 1.
- No files available in the specified path.
- Unable to identify the input file as either Suspect or Candidate File.
- ExpressMatchKey defined but not available for the record\t
- Unable to get the FileName of the InputSplit.
- Unable to initialize engine.
- Error processing consolidated records.

### *Exceptions - Hive User-Defined Functions*

- _FUNC_ must have the minimum arguments.
- Unable to initialize engine. Rule passed: `<Rule used>`
- Expected argument type: String. Received argument type: `<Mismatched Type>`
- Exception: `<Header string>` configuration missing.
- Error processing consolidated records: `<Exception details>`
- Exception: Sort field column `<column name>` missing from job configuration.

# B - Enums

## In this section

# Common Enumerations

### *Enum* `MatchingAlgorithm`

Package: `com.pb.bdq.api.matcher`

Class: `Algorithm`

1. Acronym
2. CharacterFrequency
3. DaitchMokotoffSoundex
4. Date
5. DoubleMetaphone
6. EditDistance
7. EuclideanDistance
8. ExactMatch
9. Initials
10. JaroWinklerDistance
11. KeyboardDistance
12. Koeln
13. KullbackLeiblerDistance
14. Metaphone
15. SpanishMetaphone
16. Metaphone3
17. NGramDistance
18. NGramSimilarity
19. NumericString
20. Nysiis
21. Phonix
22. Soundex
23. SubString
24. SyllableAlignment

### *Enum* `Algorithm`

Package: `com.pb.bdq.api.matchkeygenerator`

Class: `MatchKeyRule`

1. Soundex
2. Metaphone
3. SpanishMetaphone
4. DoubleMetaphone
5. Nysiis

6. Phonix
7. Metaphone3
8. Koeln
9. Consonant
10. SubString

### *Enum* `RecordSeparator`
Package: `com.pb.bdq.common.job`

Class: `FilePath`

1. WINDOWS
2. LINUX
3. MACINTOSH

### *Enum* `ReferenceDataPathLocation`
Package: `com.pb.bdq.common.job`

| Enum Constant | Description |
|---|---|
| HDFS | The Reference Data is placed on HDFS. |
| LocaltoDataNodes | The Reference Data is placed on all available data nodes in the cluster. |

### *Enum* `Operation`
Package: `com.pb.bdq.api.consolidation`

1. CONTAINS
2. HIGHEST
3. LOWEST
4. NOT_EQUAL
5. GREATER
6. LESSER
7. EQUAL
8. GREATER_THAN_EQUAL_TO
9. LESS_THAN_EQUAL_TO
10. IS_EMPTY
11. IS_NOT_EMPTY
12. MOST_COMMON
13. LONGEST
14. SHORTEST

### *Enum* `MatchingMethod`
Package: `com.pb.bdq.api.matcher`

Class: `ParentMatchRule`

1. AllTrue
2. AnyTrue
3. BasedOnThreshold

### *Enum `ScoringMethod`*
Package: `com.pb.bdq.api.matcher`

Class: `MatchRule`

1. Minimum
2. Maximum
3. Average
4. WeightedAverage
5. VectorSummation

### *Enum `MissingDataMethod`*
Package: `com.pb.bdq.api.matcher`

Class: `MatchRule`

1. IgnoreBlanks
2. CountAs100
3. CountAs0
4. CompareBlanks

### *Enum `JoinDetail.JoinType`*
Package: `com.pb.bdq.dim.api.detai`

Class: `JoinDetail`

1. Full
2. Inner
3. LeftOuter

### *Enum `JoinType`*
Package: `com.pb.bdq.api.consolidation`

Class: `ConjoinedRule`

1. OR
2. AND

### *Enum `IncludeTerm`*
Package: `com.pb.bdq.api.advtransformer`

Class: `TableDataExtraction`

1. ExtractedData

2. NonExtractedData
3. TermNeither

### *Enum* `Extract`

Package: `com.pb.bdq.api.advtransformer`

Class: `TableDataExtraction`

1. ExtractTerm
2. ExtractNWordsLeft
3. ExtractNWordsRight

### *Enum* `AdvTransformerExtractionType`

Package: `com.pb.bdq.api.advtransformer`

Class: `AbstractAdvancedTransformerRules`

1. TableData
2. RegularExpression

### *Enum* `MatchRuleType`

Package: `com.pb.bdq.api.matcher`

Class: `MatchRule`

1. Parent
2. Child

### *Enum* `SortInput`

Package: `com.pb.bdq.api.matcher`

Class: `MatchRule`

1. CHARS
2. TERMS

### *Enum* `TableLookupAction`

Package: `com.pb.bdq.api.tablelookup`

Class: `AbstractTableLookupRule`

1. Standardize
2. Categorize
3. Identify

# Universal Addressing Enumerations

### *Enum* `DatabaseType`
Package: `com.pb.bdq.api.uam.global`

Class: `GlobalAddressingEngineConfiguration`

1. BATCH_INTERACTIVE
2. FASTCOMPLETION
3. CERTIFIED

### *Enum* `PreloadingType`
Package: `com.pb.bdq.api.uam.global`

Class: `GlobalAddressingEngineConfiguration`

1. NONE
2. FULL
3. PARTIAL

### *Enum* `CountryCodes`
Package: `com.pb.bdq.api.uam`

Description: Alphabetical codes assigned to all supported countries.

### *Enum* `StateProvinceType`
Package: `com.pb.bdq.api.uam.global`

Interface: `GlobalAddressingInputOption`

1. COUNTRY_STANDARD
2. ABBREVIATION
3. EXTENDED

### *Enum* `CountryType`
Package: `com.pb.bdq.api.uam.global`

Interface: `GlobalAddressingInputOption`

1. ISO2
2. ISO3
3. ISO_NUMBER
4. NAME_CN
5. NAME_DA
6. NAME_DE
7. NAME_EN

8. NAME_ES
9. NAME_FI
10. NAME_FR
11. NAME_GR
12. NAME_HU
13. NAME_IT
14. NAME_JP
15. NAME_KR
16. NAME_NL
17. NAME_PL
18. NAME_PT
19. NAME_RU
20. NAME_SA
21. NAME_SE

### *Enum* `PreferredScript`

Package: `com.pb.bdq.api.uam.global`

Interface: `GlobalAddressingInputOption`

1. DATABASE
2. POSTAL_ADMIN_PREF
3. POSTAL_ADMIN_ALT
4. LATIN
5. LATIN_ALT
6. ASCII_SIMPLIFIED
7. ASCII_EXTENDED

### *Enum* `PreferredLanguage`

Package: `com.pb.bdq.api.uam.global`

Interface: `GlobalAddressingInputOption`

1. DATABASE
2. ENGLISH

### *Enum* `Casing`

Package: `com.pb.bdq.api.uam.global`

Interface: `GlobalAddressingInputOption`

1. NATIVE
2. UPPER
3. LOWER
4. MIXED
5. NOCHANGE

### *Enum* `OptimizationLevel`
Package: `com.pb.bdq.api.uam.global`

Interface: `GlobalAddressingInputOption`

1. NARROW
2. STANDARD
3. WIDE

### *Enum* `Mode`
Package: `com.pb.bdq.api.uam.global`

Interface: `GlobalAddressingInputOption`

1. BATCH
2. CERTIFIED
3. FASTCOMPLETION
4. INTERACTIVE
5. PARSE

### *Enum* `MatchingScope`
Package: `com.pb.bdq.api.uam.global`

Interface: `GlobalAddressingInputOption`

1. ALL
2. LOCALITY_LEVEL
3. STREET_LEVEL
4. DELIVERYPOINT_LEVEL

### *Enum* `FormatDelimiter`
Package: `com.pb.bdq.api.uam.global`

Interface: `GlobalAddressingInputOption`

1. CRLF
2. LF
3. CR
4. SEMICOLON
5. COMMA
6. TAB
7. PIPE
8. SPACE

### *Enum* `ExhaustedAction`
Package: `com.pb.bdq.api.uam.loqate`

Class: `LoqateAddressingGeneralConfiguration`

1. GROW
2. BLOCK
3. FAIL

### *Enum* `AcceptanceLevel`

Package: `com.pb.bdq.api.uam.loqate.validate`

Class: `LoqateAddressingValidateConfiguration`

1. Level0
2. Level1
3. Level2
4. Level3
5. Level4
6. Level5

### *Enum* `OutputCasing`

Package: `com.pb.bdq.api.uam.loqate.validate`

Class: `LoqateAddressingValidateConfiguration`

1. Mixed
2. Upper

### *Enum* `CountryFormat`

Package: `com.pb.bdq.api.uam.loqate.validate`

Class: `LoqateAddressingValidateConfiguration`

1. ENGLISH
2. ISO
3. UPU

### *Enum* `ScriptAlphabet`

Package: `com.pb.bdq.api.uam.loqate.validate`

Class: `LoqateAddressingValidateConfiguration`

1. InputScript
2. Native
3. Latin_English

### *Enum* `CacheSize`

Package: `com.pb.bdq.api.uam.global`

Class: `GlobalAddressingGeneralConfiguration`

1. NONE
2. SMALL

3. LARGE

### *Enum* `RangesToExpand`
Package: `com.pb.bdq.api.uam.global`

Class: `GlobalAddressingGeneralConfiguration`

1. NONE
2. ONLY_WITH_VALID_ITEMS

### *Enum* `FlexibleRangeExpansion`
Package: `com.pb.bdq.api.uam.global`

Class: `GlobalAddressingGeneralConfiguration`

1. ON
2. OFF

### *Enum* `CasingType`
Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. MIXED
2. UPPER

### *Enum* `CityNameFormat`
Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. CITY_FORMAT_LONG
2. CITY_FORMAT_SHORT
3. CITY_FORMAT_STANDARD

### *Enum* `OutputCountryFormat`
Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. ENGLISH
2. FRENCH
3. GERMAN
4. SPANISH
5. ISO
6. UPU

### *Enum* `DualAddressLogic`
Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. DUAL_NORMAL
2. DUAL_PO_BOX
3. DUAL_STREET

### *Enum `StandardAddressFormat`*
Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. STANDARD_ADDRESS_FORMAT_COMBINED_UNIT
2. STANDARD_ADDRESS_FORMAT_SEPARATE_UNIT
3. STANDARD_ADDRESS_FORMAT_SEPARATE_DUAL

### *Enum `StreetMatchingStrictness`*
Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. MATCHING_STRICTNESS_EQUAL
2. MATCHING_STRICTNESS_TIGHT
3. MATCHING_STRICTNESS_MEDIUM
4. MATCHING_STRICTNESS_LOOSE

### *Enum `FirmMatchingStrictness`*
Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. MATCHING_STRICTNESS_EQUAL
2. MATCHING_STRICTNESS_TIGHT
3. MATCHING_STRICTNESS_MEDIUM
4. MATCHING_STRICTNESS_LOOSE

### *Enum `DirectionalMatchingStrictness`*
Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. MATCHING_STRICTNESS_EQUAL
2. MATCHING_STRICTNESS_TIGHT
3. MATCHING_STRICTNESS_MEDIUM
4. MATCHING_STRICTNESS_LOOSE

### *Enum `StandardAddressPMBLine`*
Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. STANDARD_ADDRESS_PMB_LINE_NONE
2. STANDARD_ADDRESS_PMB_LINE_1
3. STANDARD_ADDRESS_PMB_LINE_2

### *Enum* `PreferredCity`

Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. CITY_OVERRIDE_NAME_ZIP4
2. CITY_USPS_STATE_FILE
3. CITY_PRIMARY_NAME

### *Enum* `DPVFileType`

Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressGeneralConfiguration`

1. SPLIT
2. FULL
3. FLAT

### *Enum* `DPVMemoryModel`

Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressGeneralConfiguration`

1. PICO
2. MICRO
3. SMALL
4. MEDIUM
5. LARGE
6. HUGE

### *Enum* `LacsLinkMemoryModel`

Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressGeneralConfiguration`

1. PICO
2. MICRO
3. SMALL
4. MEDIUM
5. LARGE
6. HUGE

### *Enum* `SuiteLinkMemoryModel`

Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressGeneralConfiguration`

1. PICO
2. MICRO
3. SMALL
4. MEDIUM
5. LARGE
6. HUGE

### *Enum `DPVSuccessStatusCondition`*
Package: `com.pb.bdq.api.universaladdress`

Class: `UniversalAddressInputConfiguration`

1. DPV_CONDITON_FULL
2. DPV_CONDITON_PARTIAL
3. DPV_CONDITON_ALWAYS

### *Enum `UAMCASSReportType`*
Package: `com.pb.bdq.uam.common`

1. CASS_3553
2. CASS_DETAIL
3. CASS_DETAIL2
4. CASS_DETAIL3

### *Enum `AddressValidationProcessType`*
Package: `com.pb.bdq.api.uam.addressvalidation`

1. VALIDATE

### *Enum `AddressValidationInputOption.MatchMode`*
Package: `com.pb.bdq.api.uam.addressvalidation`

1. Custom
2. Exact
3. Relaxed
4. Standard

# C - ISO Country Codes and Module Support

## In this section

# ISO Country Codes and Coder Support

This table lists the two-digit and three-digit ISO codes for each country as well as the level of support for each coder: Validate Address International (VAI), Validate Address Global (VAG), and Validate Address Loqate (VAL). The levels are defined as follows:

- Level A—Street-level data is provided.
- Level B—City and/or postal code data is provided.
- Level C—Country data is provided.
- '-'—Country not supported

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Afghanistan | AF | AFG | B | B | A |
| Aland Islands | AX | ALA | B | - | A |
| Albania | AL | ALB | B | B | A |
| Algeria | DZ | DZA | B | B | A |
| American Samoa | AS | ASM | B | - | A |
| Andorra | AD | AND | A | A | A |
| Angola | AO | AGO | B | B | A |
| Anguilla | AI | AIA | B | B | B |
| Antarctica | AQ | ATA | C | B | B |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Antigua And Barbuda | AG | ATG | A | B | B |
| Argentina | AR | ARG | A | A | A |
| Armenia | AM | ARM | B | A | A |
| Aruba | AW | ABW | A | B | A |
| Australia | AU | AUS | A | A | A |
| Austria | AT | AUT | A | A | A |
| Azerbaijan | AZ | AZE | B | B | A |
| Bahamas | BS | BHS | A | B | A |
| Bahrain | BH | BHR | A | A | A |
| Bangladesh | BD | BGD | B | B | A |
| Barbados | BB | BRB | A | B | A |
| Belarus | BY | BLR | A | A | A |
| Belgium | BE | BEL | A | A | A |
| Belize | BZ | BLZ | A | B | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Benin | BJ | BEN | B | B | A |
| Bermuda | BM | BMU | A | A | A |
| Bhutan | BT | BTN | B | B | B |
| Bolivia, Plurinational State Of | BO | BOL | B | B | A |
| Bonaire, Saint Eustatius And Saba | BQ | BES | B | - | B |
| Bosnia And Herzegovina | BA | BIH | B | B | A |
| Botswana | BW | BWA | B | B | A |
| Bouvet Island | BV | BVT | C | - | - |
| Brazil | BR | BRA | A | A | A |
| British Indian Ocean Territory | IO | IOT | B | B | B |
| Brunei Darussalam | BN | BRN | A | A | A |
| Bulgaria | BG | BGR | A | A | A |
| Burkina Faso | BF | BFA | A | B | A |
| Burundi | BI | BDI | B | B | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Cambodia | KH | KHM | B | B | A |
| Cameroon | CM | CMR | B | B | A |
| Canada | CA | CAN | C | A | A |
| Cape Verde | CV | CPV | B | B | A |
| Cayman Islands | KY | CYM | A | B | A |
| Central African Republic | CF | CAF | B | B | A |
| Chad | TD | TCD | B | B | A |
| Chile | CL | CHL | A | A | A |
| China | CN | CHN | B | A | A |
| Christmas Island | CX | CXR | B | - | B |
| Cocos (Keeling) Islands | CC | CCK | B | - | B |
| Colombia | CO | COL | B | A | A |
| Comoros | KM | COM | B | B | B |
| Congo | CG | COG | B | B | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Congo, The Democratic Republic Of The | CD | COD | B | B | A |
| Cook Islands | CK | COK | A | B | B |
| Costa Rica | CR | CRI | B | B | A |
| Côte d'Ivoire | CI | CIV | B | B | A |
| Croatia | HR | HRV | A | A | A |
| Cuba | CU | CUB | A | B | A |
| Curacao | CW | CUW | B | B | B |
| Cyprus | CY | CYP | A | A | A |
| Czech Republic | CZ | CZE | A | A | A |
| Denmark | DK | DNK | A | A | A |
| Djibouti | DJ | DJI | B | B | B |
| Dominica | DM | DMA | B | B | B |
| Dominican Republic | DO | DOM | B | A | A |
| Ecuador | EC | ECU | A | B | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Egypt | EG | EGY | B | B | A |
| El Salvador | SV | SLV | A | B | A |
| Equatorial Guinea | GQ | GNQ | B | B | A |
| Eritrea | ER | ERI | B | B | A |
| Estonia | EE | EST | A | A | A |
| Ethiopia | ET | ETH | B | B | A |
| Falkland Islands (Malvinas) | FK | FLK | A | B | A |
| Faroe Islands | FO | FRO | A | B | B |
| Fiji | FJ | FJI | A | B | B |
| Finland | FI | FIN | A | A | A |
| France | FR | FRA | A | A | A |
| French Guiana | GF | GUF | A | - | A |
| French Polynesia | PF | PYF | B | - | B |
| French Southern Territories | TF | ATF | C | - | B |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Gabon | GA | GAB | B | B | A |
| Gambia | GM | GMB | B | B | A |
| Georgia | GE | GEO | B | A | A |
| Germany | DE | DEU | A | A | A |
| Ghana | GH | GHA | B | B | A |
| Gibraltar | GI | GIB | A | A | A |
| Greece | GR | GRC | B | A | A |
| Greenland | GL | GRL | B | A | B |
| Grenada | GD | GRD | B | B | B |
| Guadeloupe | GP | GLP | A | - | A |
| Guam | GU | GUM | C | - | A |
| Guatemala | GT | GTM | B | B | A |
| Guernsey | GG | GGY | C | - | A |
| Guinea | GN | GIN | B | B | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Guinea-Bissau | GW | GNB | B | B | A |
| Guyana | GY | GUY | B | B | A |
| Haiti | HT | HTI | A | A | A |
| Heard Island and McDonald Islands | HM | HMD | C | - | - |
| Holy See (Vatican City State) | VA | VAT | A | A | A |
| Honduras | HN | HND | B | B | A |
| Hong Kong | HK | HKG | A | A | A |
| Hungary | HU | HUN | A | A | A |
| Iceland | IS | ISL | A | A | A |
| India | IN | IND | A | A | A |
| Indonesia | ID | IDN | A | A | A |
| Iran, Islamic Republic Of | IR | IRN | B | B | A |
| Iraq | IQ | IRQ | B | B | A |
| Ireland | IE | IRL | A | A | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Isle Of Man | IM | IMN | C | - | A |
| Israel | IL | ISR | B | A | A |
| Italy | IT | ITA | A | A | A |
| Jamaica | JM | JAM | B | B | A |
| Japan | JP | JPN | A | A | A |
| Jersey | JE | JEY | C | - | A |
| Jordan | JO | JOR | B | B | A |
| Kazakhstan | KZ | KAZ | B | A | A |
| Kenya | KE | KEN | B | B | A |
| Kiribati | KI | KIR | B | B | B |
| Korea, Democratic People's Republic Of | KP | PRK | B | B | A |
| Korea, Republic Of | KR | KOR | B | A | A |
| Kuwait | KW | KWT | A | A | A |
| Kyrgyzstan | KG | KGZ | A | B | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Lao People's Democratic Republic | LA | LAO | B | B | A |
| Latvia | LV | LVA | A | A | A |
| Lebanon | LB | LBN | A | B | A |
| Lesotho | LS | LSO | B | B | A |
| Liberia | LR | LBR | A | B | B |
| Libyan Arab Jamahiriya | LY | LBY | B | B | B |
| Liechtenstein | LI | LIE | B | A | A |
| Lithuania | LT | LTU | A | A | A |
| Luxembourg | LU | LUX | A | A | A |
| Macao | MO | MAC | A | A | A |
| Macedonia, Former Yugoslav Republic Of | MK | MKD | A | B | A |
| Madagascar | MG | MDG | A | B | A |
| Malawi | MW | MWI | B | B | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Malaysia | MY | MYS | A | A | A |
| Maldives | MV | MDV | A | A | A |
| Mali | ML | MLI | B | B | A |
| Malta | MT | MLT | A | A | A |
| Marshall Islands | MH | MHL | C | - | A |
| Martinique | MQ | MTQ | A | - | A |
| Mauritania | MR | MRT | B | B | A |
| Mauritius | MU | MUS | A | B | A |
| Mayotte | YT | MYT | A | - | B |
| Mexico | MX | MEX | B | A | A |
| Micronesia, Federated States Of | FM | FSM | C | - | A |
| Moldova, Republic Of | MD | MDA | A | A | A |
| Monaco | MC | MCO | A | A | A |
| Mongolia | MN | MNG | B | B | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Montenegro | ME | MNE | B | B | A |
| Montserrat | MS | MSR | A | B | B |
| Morocco | MA | MAR | A | A | A |
| Mozambique | MZ | MOZ | B | B | A |
| Myanmar | MM | MMR | B | B | A |
| Namibia | NA | NAM | A | B | A |
| Nauru | NR | NRU | B | B | A |
| Nepal | NP | NPL | B | B | A |
| Netherlands | NL | NLD | A | A | A |
| New Caledonia | NC | NCL | B | - | B |
| New Zealand | NZ | NZL | A | A | A |
| Nicaragua | NI | NIC | B | B | B |
| Niger | NE | NER | A | B | A |
| Nigeria | NG | NGA | A | A | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Niue | NU | NIU | C | B | B |
| Norfolk Island | NF | NFK | B | B | B |
| Northern Mariana Islands | MP | MNP | C | - | A |
| Norway | NO | NOR | A | A | A |
| Oman | OM | OMN | B | B | A |
| Pakistan | PK | PAK | A | B | A |
| Palau | PW | PLW | C | - | A |
| Palestinian Territory, Occupied | PS | PSE | C | - | B |
| Panama | PA | PAN | B | A | A |
| Papua New Guinea | PG | PNG | B | B | B |
| Paraguay | PY | PRY | A | A | A |
| Peru | PE | PER | B | B | A |
| Philippines | PH | PHL | A | A | A |
| Pitcairn | PN | PCN | B | B | B |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Poland | PL | POL | A | A | A |
| Portugal | PT | PRT | A | A | A |
| Puerto Rico | PR | PRI | C | - | A |
| Qatar | QA | QAT | A | A | A |
| Reunion | RE | REU | A | - | A |
| Romania | RO | ROU | A | A | A |
| Russian Federation | RU | RUS | B | A | A |
| Rwanda | RW | RWA | A | B | A |
| Saint Barthelemy | BL | BLM | A | - | A |
| Saint Helena, Ascension & Tristan Da Cunha | SH | SHE | B | B | A |
| Saint Kitts and Nevis | KN | KNA | B | A | A |
| Saint Lucia | LC | LCA | B | B | A |
| Saint Martin (French Part) | MF | MAF | A | - | B |
| Saint Pierre and Miquelon | PM | SPM | B | - | B |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Saint Vincent and the Grenadines | VC | VCT | A | B | A |
| Samoa | WS | WSM | B | B | A |
| San Marino | SM | SMR | B | A | A |
| Sao Tome and Principe | ST | STP | B | B | A |
| Saudi Arabia | SA | SAU | A | A | A |
| Senegal | SN | SEN | A | B | A |
| Serbia | RS | SRB | A | A | A |
| Seychelles | SC | SYC | B | B | B |
| Sierra Leone | SL | SLE | A | B | A |
| Singapore | SG | SGP | A | A | A |
| Sint Maarten (Dutch Part) | SX | SXM | B | B | B |
| Slovakia | SK | SVK | A | A | A |
| Slovenia | SI | SVN | A | A | A |
| Solomon Islands | SB | SLB | A | B | B |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Somalia | SO | SOM | B | B | A |
| South Africa | ZA | ZAF | A | A | A |
| South Georgia And The South Sandwich Islands | GS | SGS | B | B | B |
| South Sudan | SS | SSD | B | B | A |
| Spain | ES | ESP | A | A | A |
| Sri Lanka | LK | LKA | B | B | A |
| Sudan | SD | SDN | B | B | A |
| Suriname | SR | SUR | B | B | A |
| Svalbard And Jan Mayen | SJ | SJM | A | - | B |
| Swaziland | SZ | SWZ | B | B | A |
| Sweden | SE | SWE | A | A | A |
| Switzerland | CH | CHE | A | A | A |
| Syrian Arab Republic | SY | SYR | B | B | A |
| Taiwan, Province of China | TW | TWN | A | A | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Tajikistan | TJ | TJK | B | B | B |
| Tanzania, United Republic Of | TZ | TZA | B | B | A |
| Thailand | TH | THA | B | A | A |
| Timor-Leste | TL | TLS | B | A | B |
| Togo | TG | TGO | B | B | A |
| Tokelau | TK | TKL | C | B | B |
| Tonga | TO | TON | B | B | A |
| Trinidad and Tobago | TT | TTO | B | B | A |
| Tunisia | TN | TUN | B | B | A |
| Turkey | TR | TUR | A | A | A |
| Turkmenistan | TM | TKM | B | B | B |
| Turks And Caicos Islands | TC | TCA | A | B | B |
| Tuvalu | TV | TUV | B | B | B |
| Uganda | UG | UGA | B | B | A |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Ukraine | UA | UKR | B | A | A |
| United Arab Emirates | AE | ARE | A | A | A |
| United Kingdom | GB | GBR | A | A | A |
| United States | US | USA | C | A | A |
| United States Minor Outlying Islands | UM | UMI | C | - | B |
| Uruguay | UY | URY | A | A | A |
| Uzbekistan | UZ | UZB | B | B | A |
| Vanuatu | VU | VUT | B | B | B |
| Venezuela, Bolivarian Republic Of | VE | VEN | B | A | A |
| Viet Nam | VN | VNM | B | A | A |
| Virgin Islands, British | VG | VGB | B | B | B |
| Virgin Islands, U.S. | VI | VIR | C | - | A |
| Wallis and Futuna | WF | WLF | A | - | B |

| ISO Country Name | ISO 3116-1 Alpha-2 | ISO 3116-1 Alpha-3 | VAI | VAG | VAL |
|---|---|---|---|---|---|
| Western Sahara | EH | ESH | C | B | B |
| Yemen | YE | YEM | B | B | B |
| Zambia | ZM | ZMB | A | A | A |
| Zimbabwe | ZW | ZWE | B | B | A |

# Notices

### *USPS® Notices*

Pitney Bowes Inc. holds a non-exclusive license to publish and sell ZIP + 4® databases on optical and magnetic media. The following trademarks are owned by the United States Postal Service: CASS, CASS Certified, DPV, eLOT, FASTforward, First-Class Mail, Intelligent Mail, LACS$^{Link}$, NCOA$^{Link}$, PAVE, PLANET Code, Postal Service, POSTNET, Post Office, RDI, Suite$^{Link}$ , United States Postal Service, Standard Mail, United States Post Office, USPS, ZIP Code, and ZIP + 4. This list is not exhaustive of the trademarks belonging to the Postal Service.

Pitney Bowes Inc. is a non-exclusive licensee of USPS® for NCOA$^{Link}$® processing.

Prices for Pitney Bowes Software's products, options, and services are not established, controlled, or approved by USPS® or United States Government. When utilizing RDI™ data to determine parcel-shipping costs, the business decision on which parcel delivery company to use is not made by the USPS® or United States Government.

### *Data Provider and Related Notices*

Data Products contained on this media and used within Pitney Bowes Software applications are protected by various trademarks and by one or more of the following copyrights:

© Copyright United States Postal Service. All rights reserved.

© 2014 TomTom. All rights reserved. TomTom and the TomTom logo are registered trademarks of TomTom N.V.

© 2016 HERE

Fuente: INEGI (Instituto Nacional de Estadística y Geografía)

Based upon electronic data © National Land Survey Sweden.

© Copyright United States Census Bureau

© Copyright Nova Marketing Group, Inc.

Portions of this program are © Copyright 1993-2007 by Nova Marketing Group Inc. All Rights Reserved

© Copyright Second Decimal, LLC

© Copyright Canada Post Corporation

This CD-ROM contains data from a compilation in which Canada Post Corporation is the copyright owner.

© 2007 Claritas, Inc.

The Geocode Address World data set contains data licensed from the GeoNames Project (**www.geonames.org**) provided under the Creative Commons Attribution License ("Attribution

License") located at **http://creativecommons.org/licenses/by/3.0/legalcode**. Your use of the GeoNames data (described in the Spectrum™ Technology Platform User Manual) is governed by the terms of the Attribution License, and any conflict between your agreement with Pitney Bowes Software, Inc. and the Attribution License will be resolved in favor of the Attribution License solely as it relates to your use of the GeoNames data.

3001 Summer Street
Stamford CT 06926-0700
USA

www.pitneybowes.com