pitney bowes

Big Data Quality SDK

Version 12.0

Big Data Quality SDK Guide

Table of Contents

1 - Getting Started

In this section

Introduction

The Big Data Quality 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 execute the jobs directly on a Hadoop platform, thus eliminating network delays and running distributed Data Quality processes in cluster, resulting in a drastic improvement in the performance.

The modules supported in the Big Data Quality SDK are:

1. Advanced Matching Module
2. Data Normalization Module
3. Universal Name Module
4. Universal Addressing Module

SDK Usage

This SDK can currently be used through:

1. Java APIs: Supports MapReduce and Spark
2. Hive User-Defined Functions

Reporting

The Big Data Quality SDK provides the feature of *Reporting* for certain jobs. This feature uses specific counters for each supported job, which allow you to monitor the match success achieved by the corresponding job. The various counters track the number of duplicate records, the number of unique records, and other useful parameters for an executed job.

The *Reporting* feature is currently supported in these jobs:

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

Workflow

To use the SDK, the components required are:

Big Data Quality SDK Installation
The Big Data Quality 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 Big Data Quality SDK JAR file must be imported into your Java application.

Hadoop Platform
On running a job using the Big Data Quality SDK, data is first read from the configured Hadoop platform, and after the 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 **Overview** on page 8.

Reference Data
The Reference Data, required by the Big Data Quality SDK, is placed on the Hadoop cluster.

Java API
To use the Java API, you can opt to place the reference data on either of the below:

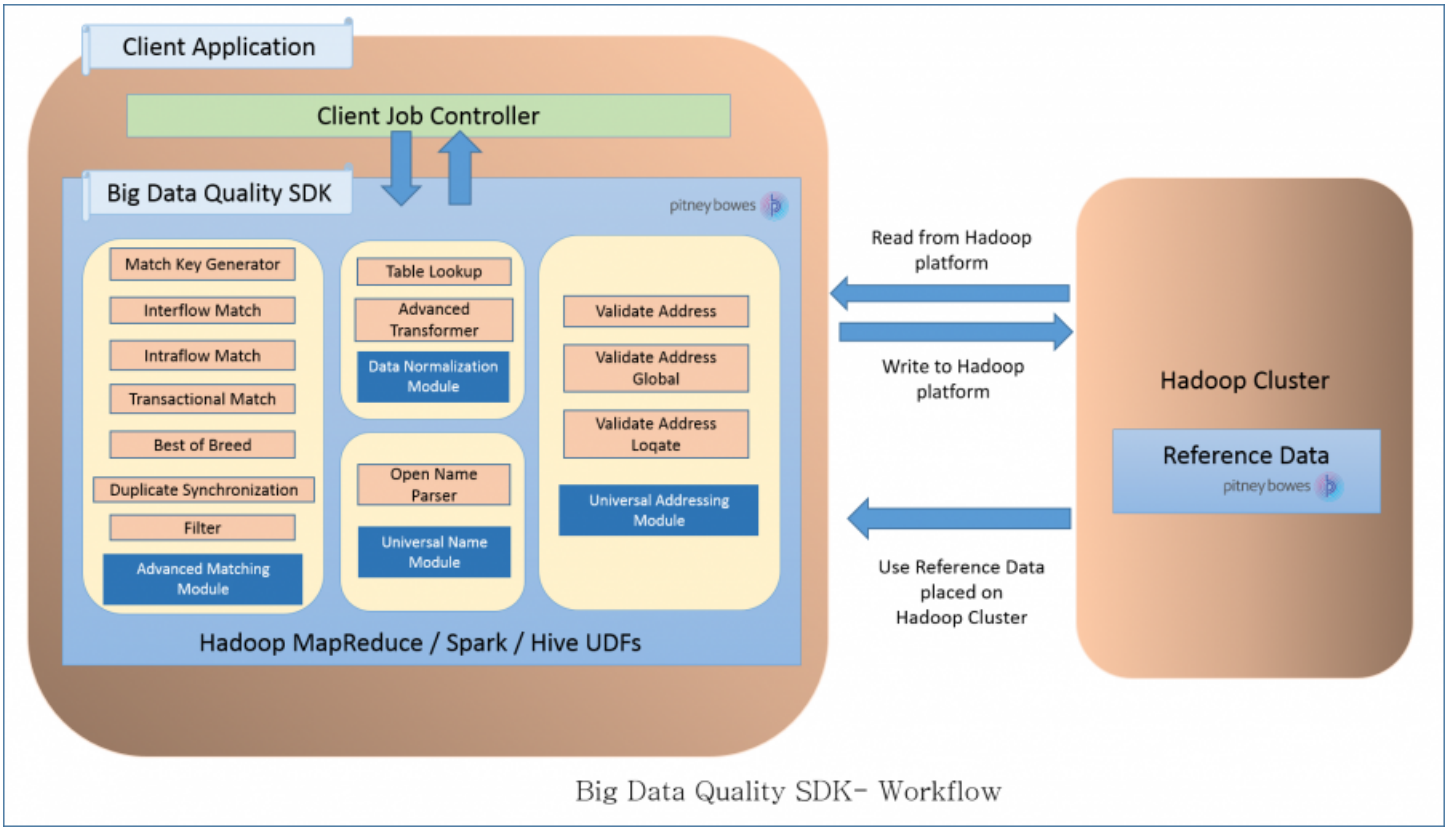
- **Local Data Nodes**: The Reference Data is placed on all available data nodes in the cluster.

  Note: This is not a failsafe method.

- **Hadoop Distributed File System (HDFS)**: The Reference Data is placed on an HDFS directory. This ensures your data is failsafe.

Hive UDFs
To use the Hive UDFs, you must place the reference data on each local data node of the cluster.

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



Big Data Quality SDK- Workflow

Who should use the SDK?

The Big Data Quality SDK is intended for:

1. Customers who want to do data quality on 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.

2 - Installation

In this section

System Requirements

For Hadoop Distributed File System (HDFS) usage:

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

For Hive usage:

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

Note: Spectrum™ Technology Platform can be run only with Hadoop clusters.

Required Operating System Updates

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

Installing the SDK

Overview

Use the link in your welcome email to download the ZIP file. A typical installer ZIP file is downloaded, named like `BigDataSDK120F0101.zip`.

Extract the contents of the downloaded ZIP file on your machine to access the installer, and run the installer which guides you through the installation process. Once installed, the SDK tool is added in your system and placed at the defined location.

You can then import the Big Data Quality SDK JAR file into your project and start accessing the APIs from your machine.

Supported Modules

Big Data Quality SDK supports the modules.

1. Advanced Matching Module
2. Data Normalization Module
3. Universal Name Module
4. Universal Addressing Module

> **Note:** You must start the Acushare service before creating the first *Validate Address* job of the Universal Addressing Module. For more information, see **Running Acushare Service** on page 11.

SDK Usage

The SDK can currently be used through:

1. Java APIs
   - MapReduce API
   - Spark API
2. Hive User-Defined Functions

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 Big Data Quality SDK on a Windows machine, follow the steps below:

1. Download the Big Data Quality SDK ZIP installer file using the download instructions contained in your welcome email or the release announcement email.
2. Extract all files from the archive to a location where you want to install Big Data Quality SDK.
3. Go to the installation directory and locate the installer named *sdkinst.exe*.
4. Double-click the file `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 Big Data Quality 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.

Attention: 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
```

6. Click **Next**.

   In the **Pre-Installation Summary** screen, review the installation information.

7. Click **Install**. The Big Data Quality SDK is installed on your computer.
8. Click **Done** to finish the installation process.
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, Universal Name Module or Universal Addressing Module, you must install the respective Reference Data for each module.

Installing SDK on Linux

To install the Big Data Quality SDK using command line on a Linux machine, follow the steps below:

1. Download the Big Data Quality SDK using the download instructions contained in your welcome email or the release announcement email.
2. Extract all files from the archive to a location on the server where you want to install the Big Data Quality SDK.
3. Change the directory to the location.
4. Ensure you have `execute` permission on the files by typing the following command:

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

5. Run this command:

```
./sdkinst.bin
```

Follow the prompts on the command prompt.

6. When prompted, provide the directory where you want to install the SDK.

   For example, `/home/hadoop/BDQ_InstallPath`.

   Attention: 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 Big Data Quality 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, 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 Big Data Quality SDK installation path to any location on the node.

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

2. Login to the node with admin rights or as a root user.

3. Go to the path where you have copied the Acushare installer script `sdkrts.bin`.
4. Ensure you have `execute` permission on the file by typing the command:
   `chmod a+x sdkrts.bin`
5. Run the installer file and follow the prompts:
   `./sdkrts.bin`
6. When prompted, either press ENTER to select the default runtime path `/root/slave_node`, or enter an absolute path of your choice.

   Important: The runtime path for Acushare must be the same on all the nodes of the cluster for the *Validate Address* job to run.

   Note: The selected path must be present on the node before specifying here.

   The Acushare service starts automatically once the installation completes successfully.
7. 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`.

Reference Data

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.

Important: For the *Validate Address* and *Vadidate Address Global* jobs, the Reference data must be placed on all the data nodes of Hadoop cluster. For the *Validate Address Loqate* job, it must be placed at one node and that further needs to be mounted to all other datanodes.

Installation Directory Structure

In the SDK installation directory, the `Utilities/dbloader` directory contains the child folders:

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

- Data Normalization Module
- Universal Name Module

> **Note:** For more information, see **Using Reference Data: Data Normalization Module and Universal Name Module** on page 13.

aq Contains:

- The `scripts/server/installdb_unc.sh` script 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 **Using Reference Data: Universal Addressing Module** on page 14.

Using Reference Data: Data Normalization Module and Universal Name Module

To use the Reference Data for **Data Normalization Module** and **Universal Name Module** you need to run the data loader script file, for example `installdb_dnm`. Executing the script file enables you to extract Reference Data to your machine.

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 Big Data Quality 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 the reference data sets are extracted and placed after download.

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

6. Specify the path for the output directory. This is the path where your input data will be extracted to.
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.

   Note: Repeat the steps for each stage.

Using Reference Data: Universal Addressing Module

To access and use the Reference Data, first fetch the data from the e-store in ZIP format.

For *Validate Address Global* and *Validate Address Loqate*, simply extract the contents of the ZIP file and the Reference Data is ready for use.

For *Validate Address*, perform the mentioned steps to extract the Reference Data to your machine.

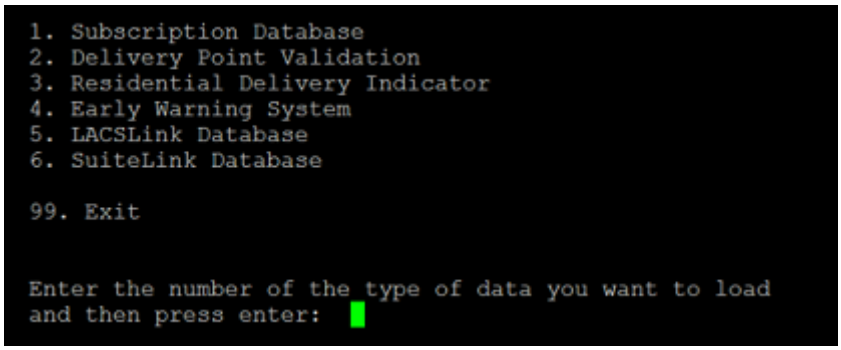
Note: Ensure that `execute` permission is granted to the `aq` folder.

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. If not, then this command starts the service.

4. After executing this command, the options displayed are:
   - **US Subscription**: Press 1 to list the available types of data loading, as mentioned in the next step.
   - **Exit**: Press 99 to exit.
5. Enter the specific number for the type of data you want to load.

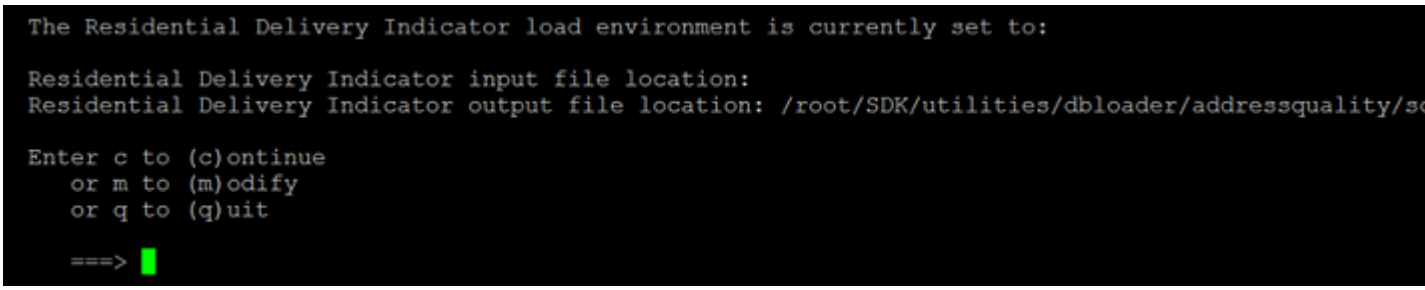


6. Specify the path where the sourced data sets are placed.

   The data sourced from the e-store is available as Reference Data input, which is required to perform the Universal Addressing Module's jobs. For the output file location, the system displays the default output path.

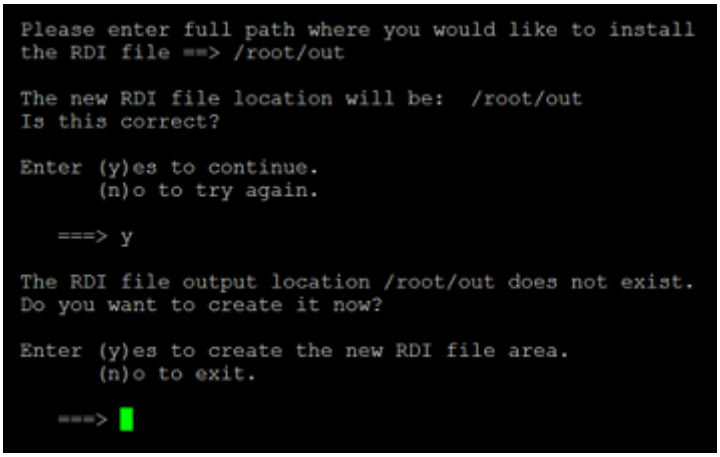
7. The input file location and the output file location are displayed.

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



   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`.



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

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

3 - Modules

In this section

Advanced Matching Module

The Advanced Matching Module matches records between and/or within any number of input files. You can also use the Advanced Matching Module to match on a variety of fields including name, address, name and address, or non-name/address fields, such as social security number or date of birth.

The Module also provides jobs to consolidate the records of a group by selecting a best record using an appropriate configuration, or by synchronizing all the records of a certain group, or filtering out a particular record from a group of records.

Supported Jobs

The Advanced Matching Module of the Big Data Quality SDK supports the jobs:

1. Match Key Generator
2. Interflow Match
   - By generating a match key
   - By using the existing match key through the Group By options
3. Intraflow Match
   - By generating a match key
   - By using the existing match key through the Group By options
4. Transactional Match
   - By generating a match key
   - By using the existing match key through the Group By options
5. Best of Breed
6. Duplicate Synchronization
7. Filter

Note: While using the Group By option, the match key is already present in the input file, using which the Group By operation is performed.

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

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. 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. 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

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. 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. 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. 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

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. 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.

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.

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.

Filter

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

Data Normalization Module

The Data Normalization Module examines terms in a record and determines if the term is in the preferred form.

- **Table Lookup**—This stage evaluates a term and compares it to a previously validated form of that term. If the term is not in the proper form, then the standard version replaces the term. Table Lookup includes changing full words to abbreviations, changing abbreviations to full words, changing nick names to full names or misspellings to corrected spellings.
- **Advanced Transformer**—This stage scans and splits strings of data into multiple fields, placing the extracted and non extracted data into an existing filed or a new field.

Supported Jobs

The Data Normalization Module of the Big Data Quality SDK supports the jobs:

1. Table Lookup
   - Table Lookup with Standardize option
   - Table Lookup with Identify option
   - Table Lookup with Categorize option
2. Advanced Transformer
   - Advanced Transformer with Table Data Extraction option
   - Advanced Transformer with Regular Expression Extraction option

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.

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

Universal Addressing Module

The Universal Addressing Module is an address quality module that can standardize and validate addresses, improving the deliverability of mail. The Universal Addressing Module can ensure that your address data adheres to quality standards established by the postal authority. An address that adheres to these standards is more likely to be delivered in a timely manner. In addition, mailers who follow these standards can qualify for significant postage discounts. For information on discounts for U.S. mail, refer to the USPS Domestic Mail Manual (DMM) available at **www.usps.com**.

> **Note:** For the UAM jobs, reference data must be placed only on local data nodes in the cluster.

Supported Jobs

The Universal Addressing Module of the Big Data Quality SDK supports the jobs:

1. Validate Address

   > **Note:** This job currently supports US address validations only.

2. Validate Address Global
3. Validate Address Loqate

Validate Address

Validate Address standardizes and validates addresses using postal authority address data. Validate Address 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.

Validate Address also returns result indicators about validation attempts, such as whether or not Validate Address 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 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, 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: Currently, Validate Address supports only US addresses.

CASS Reports

You can create and run the Validate Address job in the CASS Certified™ mode using the Big Data Quality SDK.

Additionally, you can opt to generate these types of CASS reports:

1. CASS Report 3553
2. CASS Detailed Report
3. Validate Address Summary Report

CASS Certified Processing

CASS Certified™ processing also generates the USPS CASS Detailed Report, which contains some of the same information as the 3553 report but provides much greater detail about DPV, LACS, and SuiteLink statistics. The USPS CASS Detailed Report is not required for postal discounts and does not need to be submitted with your mailing.

The CASS Detailed Report is generated in three parts, named as follows:

1. *CASS Detail*
2. *CASS Detail 2*
3. *CASS Detail 3*

For more information about the CASS settings while using the SDK, see **Using a Validate Address MapReduce Job** on page 107 and **Using a Validate Address Spark Job** on page 109. For instructions on how to use reports, see the *Dataflow Designer Guide*.

CASS 3553 Report

The USPS CASS 3553 report must be given to the USPS along with the mailing to qualify for certain discounts. The report contains information about the software you used for CASS processing, information about your name-and-address list, information about your output file, information about the mailer, and other statistics about your mailing. For detailed information about USPS Form 3553, see **www.usps.com**.

For instructions on how to use reports, see the *Dataflow Designer Guide*.

CASS Detailed Report

The USPS CASS Detailed Report does not need to be given to the USPS to qualify for certain discounts. This report contains some of the same information as the 3553 report but provides much greater detail about DPV, LACS, and SuiteLink statistics.

For instructions on how to use reports, see the *Dataflow Designer Guide*.

Validate Address Summary Report

The Validate Address Summary Report lists statistics about the job, such as the total number of records processed, the number of addresses validated, and more.

For instructions on how to use reports, see the *Dataflow Designer Guide*.

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.

Reporting Counters

The Validate Address Global job allows you to monitor the statistics of the job once the execution is complete. The counters provide the reporting statistics across all supported countries in which a particular Validate Address Global job is run.

For a list of supported countries, refer to **ISO Country Codes and Module Support** on page 198.

Country based Counters

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

Similarly, the same counters are listed for all the supported countries for which the Validate Address Global job is run.

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.

1. SUMMARY_STATUS_I4_COUNT
2. SUMMARY_STATUS_I2_COUNT
3. SUMMARY_END_TIME
4. SUMMARY_START_TIME
5. SUMMARY_STATUS_V_COUNT
6. SUMMARY_STATUS_C_COUNT
7. SUMMARY_CHARSET
8. SUMMARY_DEFAULT_COUNTRY
9. SUMMARY_STATUS_I3_COUNT
10. SUMMARY_STATUS_S_COUNT
11. SUMMARY_FAILED_COUNT
12. COUNTRY: A comma-separated list of the country codes for which the address validation is run.
13. 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*.

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, and so on.

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.

Reporting Counters

The Validate Address Loqate job allows you to monitor the results of the job. The counters available are:

1. Original Postal Code Confirmed via Address Match
2. Total Records Successfully Matched
3. House Mismatch
4. Total Records for which Address Validation Attempted
5. Input Record Count
6. Number Range Mismatch
7. Total Records Valid on Input
8. No Postal Code Available
9. Total Unmatched Recorded
10. Total Corrected
11. Total Unmatched Records
12. Postal Code Corrected via Address Match
13. Standard Address Returned Successfully
14. Address Records Processed
15. Street Mismatch
16. Original Postal Code Retained
17. Records Processed by LOQATE

Universal Name Module

To perform the most accurate standardization you may need to break up strings of data into multiple fields. The Big Data Quality SDK provides advanced parsing features that enable you to parse personal names, company names, and many other terms and abbreviations.

Supported Jobs

The Universal Name Module of the Big Data Quality SDK supports the job:

1. Open Name Parser

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.

Reporting

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. 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.

4 - The Java API

In this section

Introduction

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 usng 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 Big Data Quality SDK job using the Java API are:

JAR Files

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

Module	Job	JAR File
Advanced Matching Module	All AMM jobs	*amm.core-12.0.jar*
Data Normalization Module	All DNM jobs	*dnm.core-12.0.jar*
Universal Addressing Module	Validate Address	*uam-universaladdress.core-12.0.jar*
Universal Addressing Module	Validate Address Global	*uam-global.core-12.0.jar*
Universal Addressing Module	Validate Address Loqate	*uam-loqate.core-12.0.jar*
Universal Name Module	All UNM jobs	*unm.core-12.0.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, MapReduce or Spark configuration details, and the like. 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 Big Data Quality 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 SDK

The SDK can be used to run Big Data Quality 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 33.

2. Create your own Java client project by importing the relevant Big Data Quality 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 35.

Using Configuration Property Files

Ensure the Big Data Quality SDK is installed on your machine.

You can run a Big Data Quality SDK job using the module-specific JAR files and the configuration files in XML formats.

The sample configuration properties are shipped with the Big Data Quality SDK and are placed at the location `<Big Data Quality bundle>\samples\configuration`.

> **Note:** For a list of the module-specific JAR files, see **Components of the SDK Java API** on page 32.

1. For a Linux system, open a command prompt.

   For Windows and Unix systems, open an SSH client like 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.12.0.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.12.0.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
```

Creating a Java Application

Ensure the Big Data Quality 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 Big Data Quality 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 32.
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 39.

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 or output text file to run a job.

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

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

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 36.

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 below:

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>`. For example, `malformedRecordsCandidate-m-00000`, `malformedRecordsCandidate-m-00001`. 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>`. For example, `malformedRecordsSuspect-m-00000`, `malformedRecordsSuspect-m-00001`. 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>`. For example, `malformedRecords-m-00000`, `malformedRecords-m-00001`. 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.

Match Key Generator

Overview

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:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** Flag to indicate if header file is to be created on the Hadoop server or not.
Job Configurations	The Hadoop configurations for the job. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.
Match Key Settings	A combination of the columns and the algorithms to be applied to generate the match key, required to perform the matching. **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, the following 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. **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 38.
   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 37.
   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.
   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.
   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 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 **SparkProcessType** on page 38.
   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 **SparkJobConfig** on page 37.
   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.
   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.
   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.

Interflow Match

Overview

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 the arguments: **GroupBy Column** The name of the column using which the records are to be grouped. **Number of Reducer Tasks** The number of reducer tasks required to group the records. For a *Spark* job, to create a Group-By option pass the arguments: **GroupBy Column** The name of the column using which the records are to be grouped.
Match Rule	Define as many parent and child rules as required, to create a `MatchRule` object. For more information, see **MatchRule** on page 40.

Parameter	Description
Candidate File	*For text files:* **File Path** The path of the candidate text file on the Hadoop platform. **Record Separator** The record separator used in the candidate file. **Field Separator** The separator used between any two consecutive fields of a record, in the candidate file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the candidate file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the suspect file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. **Important:** The suspect and candidate files must be of the same format. Either both must be text files, or both must be ORC format files. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the suspect text file on the Hadoop platform. **Record Separator** The record separator used in the suspect file. **Field Separator** The separator used between any two consecutive fields of a record, in the suspect file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the suspect file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the suspect file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** Flag to indicate if header file is to be created on the Hadoop server or not.
Job Configurations	The Hadoop configurations for the job. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.
Match Key Settings	A combination of the columns and the algorithms to be applied to generate the match key, required to perform the matching. **Note:** Specify only one match key. **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.
Setting Collection Number Zero to Unique Records	Set this to `true` to set the collection number of unique records as 0 (zero).

Parameter	Description
Comparison Option	Allows you 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.
Compress Output	Flag to indicate if the output must be compressed. 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`. 2. For a duplicate candidate record matched, but not using an express match key, the output value is blank. 3. For a unique candidate record matched using an express match key, the output value is `N`. 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. **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 38.
   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 40 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 37.
   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.

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

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 46 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 post a 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 38.
   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 40 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 37.
   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.

      Important: The suspect and candidate files must be of the same format. Either both must be text files, or both must be 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.
   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 46 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

Overview

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 the arguments: **GroupBy Column** The name of the column using which the records are to be grouped. **Number of Reducer Tasks** The number of reducer tasks required to group the records. For a *Spark* job, to create a Group-By option pass the arguments: **GroupBy Column** The name of the column using which the records are to be grouped.
Match Rule	Define as many parent and child rules as required, to create a `MatchRule` object. For more information, see **MatchRule** on page 40.

Parameter	Description
Input File	*For text files:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** Flag to indicate if header file is to be created on the Hadoop server or not.
Job Configurations	The Hadoop configurations for the job. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.
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. 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. **Note:** Specify only one match key. **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`. 2. For a duplicate candidate record matched, but not using an express match key, the output value is blank. 3. For a unique candidate record matched using an express match key, the output value is blank. 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. **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 38.
   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 40 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 37.
   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.
   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.
   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 post a 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 38.
   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 40 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 37.
   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.
   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.
   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.

Transactional Match

Overview

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 the arguments: **GroupBy Column** The name of the column using which the records are to be grouped. **Number of Reducer Tasks** The number of reducer tasks required to group the records. For a *Spark* job, to create a Group-By option pass the arguments: **GroupBy Column** The name of the column using which the records are to be grouped.
Match Rule	Define as many parent and child rules as required, to create a `MatchRule` object. For more information, see **MatchRule** on page 40.

Parameter	Description
Input File	*For text files:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** 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. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.
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. Set this to `true` to compress the output.
Match Key Settings	A combination of the columns and the algorithms to be applied to generate the match key, required to perform the matching. **Note:** Specify only one match key. **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. **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: • `Y` (if duplicates are present) OR • `N` (if duplicates are absent) For Duplicate records, the output value is `D`. 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 38.
   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 40 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 37.
   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.

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.

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 post a 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 38.

   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 40 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 37.

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.

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.

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.

Best of Breed

Overview

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. For a *MapReduce* job, pass the arguments: **GroupBy Column** The name of the column using which the records are to be grouped. **Number of Reducer Tasks** The number of reducer tasks required to group the records. For a *Spark* job, pass the arguments: **GroupBy Column** 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:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** 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. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.
Compress Output	Flag to indicate if the output must be compressed. 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: **Primary** If the record is the selected template record in a collection. **Secondary** If the record is not the selected template record in a collection. **BestOfBreed** If the record is the newly created best of breed record in the collection. 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 38.
   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 40 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 187 and **Enum Operation** on page 186.

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 37.

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.

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.

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 post a 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 38.

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 40 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 187 and **Enum Operation** on page 186.

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 37.

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.

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.

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

Overview

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. For a *MapReduce* job, pass the arguments: **GroupBy Column** The name of the column using which the records are to be grouped. **Number of Reducer Tasks** The number of reducer tasks required to group the records. For a *Spark* job, to create a Group-By option pass the arguments: **GroupBy Column** The name of the column using which the records are to be grouped. **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:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** 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. 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 38.
   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 40 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 187 and **Enum Operation** on page 186.

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 37.

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.

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.

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 post a 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 38.
   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 40 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 187 and **Enum Operation** on page 186.

   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 37.

   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.

   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.

   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

Overview

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. For a *MapReduce* job, pass the arguments: **GroupBy Column** The name of the column using which the records are to be grouped. **Number of Reducer Tasks** The number of reducer tasks required to group the records. For a *Spark* job, to create a Group-By option pass the arguments: **GroupBy Column** The name of the column using which the records are to be grouped. **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:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** 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. 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 38.
   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 40 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 187 and **Enum Operation** on page 186.

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 37.

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.

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.

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 post a 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 38.

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 40 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 187 and **Enum Operation** on page 186.

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 37.

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.

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.

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.

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.

Table Lookup

Overview

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. 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. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.

Parameter	Description
Input File	*For text files:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** 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. 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. The name of the column is as entered by you. **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 38.
   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 186.
   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 37.
   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.
   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.

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 38.

   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 186.

   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 37.

   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.

   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.

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.

Advanced Transformer

Overview

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. 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. 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. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.

Parameter	Description
Input File	*For text files:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** 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. The name of the column is as entered by you. **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.

Column	Description	Output Value
Extracted Data	This output column is added if a new column name, not present in the input, is specified as the Extracted Data column. The name of the column is as entered by you. **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` and `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 38.
   a) Configure the advanced transfomer 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 transfomer 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 186.
   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 37.
   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.
   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.

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 38.
   a) Configure the advanced transfomer 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 transfomer 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 186.
   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 37.

   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.

   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.

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.

Universal Addressing Module Jobs

Common Module API

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

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: **1.** DPV Database Path **2.** Suite Link DB Path **3.** EWS Database Path **4.** RDI Database Path **5.** Lacs Database Path **6.** Reference Data Path **7.** COBOL Runtime Path **8.** Modules directory

Parameter	Description
Universal Address Validate Input Configuration	

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

Parameter	Description
	46. Output VeriMove Block **47.** DPV Determine No Stat **48.** DPV Determine Vacancy **49.** Output Abbreviated Alias **50.** Output Preferred Alias **51.** Output Preferred City **52.** Perform Suite Link **53.** Suppress Zplus Phantom Carrier R777
Universal Address General Configuration	To set JVM configurations: **1.** DPV File Type **2.** DPV Memory Model **3.** Lacs Link Memory Model **4.** Suite Link Memory Model
Job Configurations	The Hadoop configurations for the job. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.

Parameter	Description
Input File	*For text files:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** 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. 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. The CASS reports are generated in PDF format. The parameters are: **Counters** — A Map of the counters to be included in the CASS report. **Job Name** — The name of the job. This is included in the filename of the CASS report. **Path** — The directory where the created CASS report is placed. This is an optional input value for the CASS reports. 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. **Report Type** — The type of CASS report to be generated. You can specify one or more values from **Enum UAMCASSReportType** on page 196.

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

12. DPVFootnote
13. DPVVacant
14. DPVNoStat
15. SuiteLinkReturnCode
16. SuiteLinkMatchCode
17. SuiteLinkFidelity
18. 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 11.

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 38. 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 194, **Enum CasingType** on page 193, **Enum CityNameFormat** on page 193, **Enum OutputCountryFormat** on page 193, **Enum StandardAddressFormat** on page 193, **Enum StandardAddressPMBLine** on page 194, **Enum StreetMatchingStrictness** on page 194, **Enum FirmMatchingStrictness** on page 194, **Enum DirectionalMatchingStrictness** on page 194, **Enum DualAddressLogic** on page 193, and **Enum DPVSuccessStatusCondition** on page 195 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 `LocalReferenceDataPath`.
   c) To configure the various job run settings, create an instance of `UAMUSAddressingEngineConfiguration` by passing the `LocalReferenceDataPath` 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 194, **Enum DPVMemoryModel** on page 195, **Enum LacsLinkMemoryModel** on page 195, and **Enum SuiteLinkMemoryModel** on page 195.

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 37.

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

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

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 196. 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 11.

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 38. 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 194, **Enum CasingType** on page 193, **Enum CityNameFormat** on page 193, **Enum OutputCountryFormat** on page 193, **Enum StandardAddressFormat** on page 193, **Enum StandardAddressPMBLine** on page 194, **Enum StreetMatchingStrictness** on page 194, **Enum FirmMatchingStrictness** on page 194, **Enum DirectionalMatchingStrictness** on page 194, **Enum DualAddressLogic** on page 193, and **Enum DPVSuccessStatusCondition** on page 195 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 `LocalReferenceDataPath`.
   c) To configure the various job run settings, create an instance of `UAMUSAddressingEngineConfiguration` by passing the `LocalReferenceDataPath` 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 194, **Enum DPVMemoryModel** on page 195, **Enum LacsLinkMemoryModel** on page 195, and **Enum SuiteLinkMemoryModel** on page 195.

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 37.

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

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

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 196. You can specify one or more report types in this parameter.

Validate Address Global

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: **1.** Database Type **2.** Preloading Type **3.** Reference Data Path **4.** If all countries are supported. If not, list of supported Countries

Parameter	Description
Validate Address Global Input Configuration	To configure these settings for the input: **1.** State Province Type in result **2.** Matching Scope in process **3.** Force Country ISO3 in input **4.** Default Country ISO3 in input **5.** Format Delimiter in input **6.** Format Delimiter in result **7.** Include inputs in result **8.** Country Type in result **9.** Optimization Level of process **10.** Preferred Language of result **11.** Mode of process **12.** Preferred Script in result **13.** Maximum Results **14.** Casing of result
Validate Address Global General Configuration	To set JVM configurations: **1.** Cache Size **2.** Maximum Thread Count **3.** Maximum Address Object Count **4.** Ranges to expand **5.** Flexible Range Expansion **6.** Enable Transaction Logging **7.** Maximum Memory Usage in MB
Unlock Code	To unlock the data in the database.
Reference Data Path	To specify the Reference Data path details. **Note:** For the UAM jobs, reference data must be placed only on local data nodes in the cluster.
Job Configurations	The Hadoop configurations for the job. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.

Parameter	Description
Input File	*For text files:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** 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 38. For this, the steps are:
   a) Configure the JVM initialization settings by creating an instance of `GlobalAddressingGeneralConfiguration`.

      Use the enums **Enum CacheSize** on page 192, **Enum RangesToExpand** on page 192, and **Enum FlexibleRangeExpansion** on page 192.

   b) Set the details of the Reference Data path by creating an instance of `LocalReferenceDataPath`.
   c) Configure the necessary database settings by creating an instance of `GlobalAddressingEngineConfiguration` by passing the above `LocalReferenceDataPath` instance as an argument.

      1. Set the *preloading type* in this instance using the enum **Enum PreloadingType** on page 189.
      2. Set the *database type* using the **Enum DatabaseType** on page 188.
      3. Set the supported countries using the **Enum CountryCodes** on page 189.

4. If all countries are supported, set the `isAllCountries` attribute to true. Else, specify the comma-separated list of **Enum CountryCodes** on page 189 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 189, **Enum StateProvinceType** on page 189, **Enum CountryType** on page 189, **Enum PreferredScript** on page 190, **Enum PreferredLanguage** on page 190, **Enum Casing** on page 190, **Enum OptimizationLevel** on page 190, **Enum Mode** on page 190, and **Enum MatchingScope** on page 191 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 `JobConfig`, the `List` of unlock code values, the `GlobalAddressingEngineConfiguration` instance, and the `GlobalAddressingInputConfiguration` instance created earlier as the arguments to its constructor.

The `JobConfig` parameter must be an instance of type **MRJobConfig** on page 37.

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.

   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.

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

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 38. For this, the steps are:
   a) Configure the JVM initialization settings by creating an instance of `GlobalAddressingGeneralConfiguration`.

      Use the enums **Enum CacheSize** on page 192, **Enum RangesToExpand** on page 192, and **Enum FlexibleRangeExpansion** on page 192.

   b) Set the details of the Reference Data path by creating an instance of `LocalReferenceDataPath`.
   c) Configure the necessary database settings by creating an instance of `GlobalAddressingEngineConfiguration` by passing the above `LocalReferenceDataPath` instance as an argument.

      1. Set the *preloading type* in this instance using the enum **Enum PreloadingType** on page 189.
      2. Set the *database type* using the **Enum DatabaseType** on page 188.
      3. Set the supported countries using the **Enum CountryCodes** on page 189.
      4. If all countries are supported, set the `isAllCountries` attribute to true. Else, specify the comma-separated list of **Enum CountryCodes** on page 189 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 189, **Enum StateProvinceType** on page 189, **Enum CountryType** on page 189, **Enum PreferredScript** on page 190, **Enum PreferredLanguage** on page 190, **Enum Casing** on page 190, **Enum OptimizationLevel** on page 190, **Enum Mode** on page 190, and **Enum MatchingScope** on page 191 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 `JobConfig`, the `List` of unlock code values, the `GlobalAddressingEngineConfiguration` instance, and the `GlobalAddressingInputConfiguration` instance created earlier as the arguments to its constructor.

      The `JobConfig` parameter must be an instance of type **SparkJobConfig** on page 37.

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.

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

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: **1.** Verbose **2.** Tool Info **3.** Output Address Format **4.** Log Input **5.** Log Output **6.** Log File Name **7.** Match Score Absolute Threshold **8.** Match Score Threshold Factor **9.** Postal Code Max Results **10.** Strict Reference Match

Parameter	Description
Validate Address Loqate Validate Configuration	To configure these settings for the input: **1.** Include Standard Address **2.** Include Matched Address Elements **3.** Standardized Input Address Elements **4.** Return Address Data Blocks **5.** Output Casing **6.** Include Result Codes for Individual Fields **7.** Return Multiple Addresses **8.** Failed On Multi Match Found **9.** Multiple Address Count **10.** Country Format **11.** Default Country **12.** Script Alphabet **13.** Return Geocoded Address Fields **14.** Acceptance Level **15.** Minimum Match Score **16.** Format Data Using AMAS Conventions **17.** Is Duplicate Handling **18.** Single Field Duplicate Handling **19.** Multi Field Duplicate Handling **20.** Non Standard Field Duplicate Handling **21.** Output Field Duplicate Handling
Validate Address Loqate General Configuration	To set JVM configurations: **1.** Maximum Idle Objects **2.** Minimum Idle Objects **3.** Maximum Active Objects **4.** Maximum Wait Time **5.** Action When Exhausted **6.** Test on Borrow **7.** Test on Return **8.** Test While Idle **9.** Time Between Eviction Runs in Milliseconds **10.** Number of Tests Per Eviction Run **11.** Min Evictable Idle Time in Milliseconds
Reference Data Path	To specify the Reference Data path details. **Note:** For the UAM jobs, reference data must be placed only on local data nodes in the cluster.

Parameter	Description
Job Configurations	The Hadoop configurations for the job. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.
Input File	*For text files:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** 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 1: City/Street/Postal Code Centroid Match Codes

Element	Match Code
Address Point	P4
Address Point Interpolated	I4
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 38. For this, the steps are:
   a) Configure the JVM initialization settings by creating an instance of `LoqateAddressingGeneralConfiguration`.

      Use the enum **Enum ExhaustedAction** on page 191.

   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 191, **Enum CountryCodes** on page 189, **Enum OutputCasing** on page 192, **Enum CountryFormat** on page 192, and **Enum ScriptAlphabet** on page 192.

   d) Set the details of the Reference Data path by creating an instance of `LocalReferenceDataPath`.
   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 37.

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

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

      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 38. For this, the steps are:
   a) Configure the JVM initialization settings by creating an instance of `LoqateAddressingGeneralConfiguration`.
      Use the enum **Enum ExhaustedAction** on page 191.
   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 191, **Enum CountryCodes** on page 189, **Enum OutputCasing** on page 192, **Enum CountryFormat** on page 192, and **Enum ScriptAlphabet** on page 192.
   d) Set the details of the Reference Data path by creating an instance of `LocalReferenceDataPath`.
   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 37.
      1. Set the details of the input file using the `inputPath` field of the `LoqateAddressingDetail` 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.
      2. Set the details of the output file using the `outputPath` field of the `LoqateAddressingDetail` 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.

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

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. For a MapReduce job, the instance must be of type **MRJobConfig** on page 37. For a Spark job, the instance must be of type **SparkJobConfig** on page 37.

Parameter	Description
Input File	*For text files:* **File Path** The path of the input text file on the Hadoop platform. **Record Separator** The record separator used in the input file. **Field Separator** The separator used between any two consecutive fields of a record, in the input file. **Text Qualifier** The character used to surround text values in a delimited file. **Header Row Fields** An array of the header fields of the input file. **Skip First Row** Flag to indicate if the first row must be skipped while reading the input file records. This must be `true` in case the first row is a header row. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the input ORC format file on the Hadoop platform. *Common parameters:* **Field Mappings** 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:* **File Path** The path of the output text file on the Hadoop platform. **Field Separator** The separator used between any two consecutive fields of a record, in the output file. **Attention:** Invoke the appropriate constructor of `FilePath`. *For ORC format files:* **ORC File Path** The path of the output ORC format file on the Hadoop platform. *Common parameters:* **Overwrite** Flag to indicate if output file must overwrite any existing file of same name. **Create Output Header** 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		
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).

	Format	Description
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. **Null (empty)** Global culture (default). **de** German. **es** Spanish. **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.
IsPersonal	String	Indicates whether the name is an individual rather than a firm. Values are true or false.

	Format	Description
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 "&".
Conjunction3	String	Indicates that a third, conjoined name contains a conjunction such as "and", "or", or "&".

	Format	Description
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.
MiddleName2	String	The middle name of a second, conjoined name.
MiddleName3	String	The middle name of a third, conjoined name.

	Format	Description
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 38.
   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 186.
   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 37.
   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.
   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.
   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 38.
   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 186.
   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 37.
   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.
   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.
   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 - 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 Big Data Quality 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:

- Match Key Generator
- Table Lookup
- Advanced Transformer
- Open Name Parser

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:

- Interflow Match
- Intraflow Match
- Transactional Match
- Best of Breed
- Duplicate Synchronization
- Filter
- Validate Address
- Validate Address Global
- Validate Address Loqate

Components of a **Big Data Quality SDK** Hive Function

The key components required to run a Big Data Quality SDK Hive UDF are:

JAR File	The Big Data Quality 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.
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.Map.Aggr	To turn the aggregation of data between Mapper and Reducer on or off, set this Hive environment variable to `false`. By default, `Hive.Map.Aggr = true` and the data is aggregated. Set this value to `false` for all Hive jobs in the SDK. **Note:** This configuration is required for all UDAFs.
General Configurations	The memory configurations required to run the job. **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 like 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 UDAF.

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 the below 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-based 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 Big Data Quality SDK Module to which the desired Data Quality Hive UDF belongs.
3. In case of the Validate Address UDAF, to set the path of the COBOL libraries, set the environment variable `LD_LIBRARY_PATH` as below:

```
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 UDAF, add the file *libAddressDoctor5.so* file as well.
5. In case of the Validate Address Loqate UDAF, 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 configurations like 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: 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.

8. 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: Ensure to 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.

9. Switch off the aggregatiion of data between Reducer and Mapper, by seting the `Hive.Map.Aggr` environment variable configuration to `false`, as indicated in the below example:

   ```
   set hive.map.aggr = false;
   ```

   Note: This configuration is required for all UDAFs.

10. Set the general configurations for running the job as indicated in the below 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.

11. Set the input configurations for running the job as indicated in the below 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.

12. Set the engine configurations for running the job as indicated in the below example:

```
set pb.bdq.uam.universaladdress.engine.configurations = {
"referenceData":{
"dataDir":"/home/hduser/resources/uam/universaladdress/UAM_universaladdress4.0_Feb15/",
 "referenceDataPathLocation":"LocaltoDataNodes"},
"cobolRuntimePath":"/home/hduser/tapan/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.

13. 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.

14. To run the job and display the job output on the console, write the query as indicated in the below 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`, `pb.bdq.consolidation.sort.field`, and so on.

Advanced Matching Module Functions

Match Key Generator

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 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}]}]}';

set header='businessname,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 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;


-- 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 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;

--Sample data in input table customer
--+-----------------------------------------+----------------+--+
--|           cust.businessname             | 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
+-----------------------------------+---------+----------+----------------+--+
|               businessname        | 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       |
+-----------------------------------+---------+----------+----------------+--+
```

Interflow Match

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 'pb.bdq.match.rule'

set pb.bdq.match.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
'pb.bdq.suspect.header'
set
pb.bdq.match.suspect.header=name,firstname,lastname,matchkey,middlename,recordid;

-- Set the header for candidate table using configuration property
'pb.bdq.candidate.header'
set
pb.bdq.match.candidate.header=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 pb.bdq.match.sort.field=c_id;

-- Set the express match column(optional)
set pb.bdq.match.express.column=matchkey;

-- Set sort field name to the alias used in the query, using
configuration property 'pb.bdq.match.inter.comparison'
```

```
set pb.bdq.match.inter.comparison=maxNumOfDuplicates,2;

-- Optionally, one can also set
'pb.bdq.match.inter.comparison=returnUniqueCandidates,true';

-- Set sort collection number option for unique records using
configuration property 'pb.bdq.match.unique.collectnumber.zero'
set pb.bdq.match.unique.collectnumber.zero=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 lateralview.record ["MatchRecordType"],
 lateralview.record ["MatchScore"],
 lateralview.record ["HasDuplicate"],
 lateralview.record ["CollectionNumber"],
 coalesce(lateralview.record ["ExpressMatched"], ''),
 lateralview.record ["SourceType"],
 lateralview.record ["name"],
 lateralview.record ["firstname"],
 lateralview.record ["lastname"],
 lateralview.record ["matchkey"],
 lateralview.record ["middlename"],
 lateralview.record ["recordid"]
FROM (
 SELECT interMatch(s_id, s_name, s_firstname, s_lastname, s_matchkey,
s_middlename, s_recordid, c_id,c_name, c_firstname, c_lastname,
c_matchkey, c_middlename, c_recordid) AS
 OUTPUT
 FROM (
  SELECT suspects.suspect_id AS s_id,
   suspects.NAME AS s_name,
   suspects.firstname AS s_firstname,
   suspects.lastname AS s_lastname,
   suspects.matchkey AS s_matchkey,
   suspects.middlename AS s_middlename,
   suspects.recordid AS s_recordid,
   candidates.candidate_id AS c_id,
   candidates.NAME AS c_name,
   candidates.firstname AS c_firstname,
   candidates.lastname AS c_lastname,
   candidates.matchkey AS c_matchkey,
   candidates.middlename AS c_middlename,
   candidates.recordid AS c_recordid
  FROM

     (
   SELECT rowid(*) AS suspect_id
    ,*
   FROM namedataintersuspect
   ) AS suspects LEFT JOIN
   (
```

```
    SELECT rowid(*) AS candidate_id
    ,*
    FROM namedataintercandidate
   ) AS candidates
   on suspects.matchkey = candidates.matchkey

  ) AS joinrecords
 GROUP BY joinrecords.s_matchkey
 ) AS innerResult LATERAL VIEW explode(innerResult.OUTPUT) lateralview
 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 lateralview.record ["MatchRecordType"],
 lateralview.record ["MatchScore"],
 lateralview.record ["HasDuplicate"],
 lateralview.record ["CollectionNumber"],
 coalesce(lateralview.record ["ExpressMatched"], ''),
 lateralview.record ["SourceType"],
 lateralview.record ["name"],
 lateralview.record ["firstname"],
 lateralview.record ["lastname"],
 lateralview.record ["matchkey"],
 lateralview.record ["middlename"],
 lateralview.record ["recordid"]
FROM (
 SELECT interMatch(s_id, s_name, s_firstname, s_lastname, s_matchkey,
s_middlename, s_recordid, c_id,c_name, c_firstname, c_lastname,
c_matchkey, c_middlename, c_recordid) AS
 OUTPUT
 FROM (
  SELECT suspects.suspect_id AS s_id,
   suspects.NAME AS s_name,
   suspects.firstname AS s_firstname,
   suspects.lastname AS s_lastname,
   suspects.matchkey AS s_matchkey,
   suspects.middlename AS s_middlename,
   suspects.recordid AS s_recordid,
   candidates.candidate_id AS c_id,
   candidates.NAME AS c_name,
   candidates.firstname AS c_firstname,
   candidates.lastname AS c_lastname,
   candidates.matchkey AS c_matchkey,
   candidates.middlename AS c_middlename,
   candidates.recordid AS c_recordid
  FROM
```

```
    (
   SELECT rowid(*) AS suspect_id
    ,*
   FROM namedataintersuspect
   ) AS suspects LEFT JOIN
   (
    SELECT rowid(*) AS candidate_id
    ,*
    FROM namedataintercandidate
   ) AS candidates
   on suspects.matchkey = candidates.matchkey

  ) AS joinrecords
 GROUP BY joinrecords.s_matchkey
 ) AS innerResult LATERAL VIEW explode(innerResult.OUTPUT) lateralview
 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

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 'pb.bdq.match.rule'
set pb.bdq.match.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 'pb.bdq.match.header'
set pb.bdq.match.header=firstname,lastname,matchkey,middlename,id;

-- Set the express match column (optional)
set pb.bdq.match.express.column=matchkey;

-- Set sort field name to the alias used in the query, using the
configuration property 'pb.bdq.match.sort.field'
set pb.bdq.match.sort.field=id;

-- Set sort collection number option for unique records using
configuration property 'pb.bdq.match.unique.collectnumber.zero'
set pb.bdq.match.unique.collectnumber.zero=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(
    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   |
```

```
—+————————+————————+————————+————————+————————————+————————————+————————————+————————+
```

Transactional Match

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 'pb.bdq.match.rule'
set pb.bdq.match.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 'pb.bdq.match.header'
set
pb.bdq.match.header=name,firstname,lastname,matchkey,middlename,recordid,id;

-- Set sort field name to the alias used in the query, using the
configuration property 'pb.bdq.match.sort.field'
set pb.bdq.match.sort.field=id;

-- Set sort collection number option for unique records using
configuration property 'pb.bdq.match.unique.candidate.return'. The
default value is false.
set pb.bdq.match.unique.candidate.return=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(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    |
--+----------+-------+-----------------+-----------+--------+----------+
```

Best of Breed

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 'pb.bdq.consolidation.rule'

set pb.bdq.consolidation.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 'pb.bdq.consolidation.header'
set pb.bdq.consolidation.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 'pb.bdq.consolidation.sort.field'
set pb.bdq.consolidation.sort.field=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(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

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 Duplicate Synchronization 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 dupsync as
'com.pb.bdq.amm.process.hive.consolidation.duplicatesync.DuplicateSyncUDAF';

-- 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.
```

```
-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'pb.bdq.consolidation.rule'

set pb.bdq.consolidation.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 'pb.bdq.consolidation.header'
set
pb.bdq.consolidation.header=column1,column2,column3,column4,column5,id;

-- Set sort field name to alias used in query using configuration
property 'pb.bdq.consolidation.sort.field'
set pb.bdq.consolidation.sort.field=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 (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

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 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.

CREATE TEMPORARY FUNCTION filter as
'com.pb.bdq.amm.process.hive.consolidation.filter.FilterUDAF';

-- Filter 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 'pb.bdq.consolidation.rule'
set pb.bdq.consolidation.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 'pb.bdq.consolidation.header'
set
pb.bdq.consolidation.header=column1,column2,column3,column4,column5,id;

-- Set sort field name to alias used in query using configuration
property 'pb.bdq.consolidation.sort.field'
set pb.bdq.consolidation.sort.field=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 (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|
--+----------+----------+----------+----------+----------+
```

Data Normalization Module Functions

Table Lookup

Sample Hive Script

```
-- 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 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 refdir='/home/hadoop/reference';

-- set header
set 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(${hiveconf:rule}, ${hiveconf:refdir},
${hiveconf: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:refdir},
${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 |
-+--------------------+-----------------+--------------+---------------+-------+
```

Advanced Transformer

Sample Hive Script

```
-- 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.advancetransformer.AdvanceTransformerUDF';

-- Set rule
set 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 refdir='/home/hadoop/reference/';

-- set header
set header ='AccountDescription,Address';

-- 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(${hiveconf:rule}, ${hiveconf:refdir},
${hiveconf: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(${hiveconf:rule}, ${hiveconf:refdir},
${hiveconf: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          |
```

```
+-----------------------------+-----------------+-----------------+-----------------+
```

Universal Addressing Module Functions

Validate Address

Attention: Before creating and running the first Validate Address job, ensure the Acushare service is running. For steps, see **Running Acushare Service** on page 11.

Sample Hive Script

```
-- Register Universal Addressing Module [UAM-Global] 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.UAMUSAddressingUDAF';

-- 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 hive.map.aggr = false;

-- set engine configuration
set pb.bdq.uam.universaladdress.engine.configurations={ "referenceData":{

"dataDir":"/home/hduser/resources/uam/universaladdress/UAM_universaladdress4.0_Feb15/",
 "referenceDataPathLocation":"LocaltoDataNodes"},
"cobolRuntimePath":"/home/hduser/tapan/addressquality/",
"modulesDir":"/home/hduser/tapan/addressquality/modules",
"dpvDbPath":null, "suiteLinkDBPath":null, "ewsDBPath":null,
"rdiDBPath":null, "lacsDBPath":null};

-- set input configuration
```

```
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"}}};

-- set general configuration
set pb.bdq.uam.universaladdress.general.configuration =
{"dFileType":"SPLIT", "dMemoryModel":"MEDIUM",
"lacsLinkMemoryModel":"MEDIUM", "suiteLinkMemoryModel":"MEDIUM"};

-- set reference path
set pb.bdq.reference.data.local.location=/media/New
Volume/hduser/resources/uam/universaladdress/UAM_universaladdress4.0_Feb15;

-- set process type
set pb.bdq.uam.universaladdress.process.type=VALIDATE;

-- set header
set pb.bdq.header=InputKeyValue,FirmName,AddressLine1,AddressLine2,City,
StateProvince,PostalCode,Text;

-- 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(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(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        |
+-----------+----------------+------------+-------------------------------+--------------+
```

Validate Address Global

Sample Hive Script

```
-- 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.GlobalAddressingUDAF';

set hive.map.aggr = false;

-- set engine configuration
set pb.bdq.uam.global.engine.configurations=[{ "referenceData":
{"dataDir":"/media/New Volume/hduser/resources/uam/addressDoctor/5.8.0/",
 "referenceDataPathLocation":"LocaltoDataNodes"},
"databaseType":"BATCH_INTERACTIVE", "preloadingType":"NONE",
"allCountries":false, "supportedCountries":"CAN,USA,AUS"}];

-- set input configuration
set
pb.bdq.uam.global.input.configuration={"resultStateProvinceType":"COUNTRY_STANDARD",
 "processMatchingScope":"ALL", "processEnrichmentAMAS":false,
"inputForceCountryISO3":"AUS", "inputDefaultCountryISO3":"AUS",
"inputFormatDelimiter":"CRLF", "resultFormatDelimiter":"CRLF",
"resultIncludeInputs":false, "resultCountryType":"NAME_EN",
"processOptimizationLevel":"STANDARD",
"resultPreferredLanguage":"DATABASE", "processMode":"BATCH",
"resultPreferredScript":"DATABASE", "resultMaximumResults":1,
"resultCasing":"NATIVE",
"properties":{"Result.StateProvinceType":"COUNTRY_STANDARD",
"Process.MatchingScope":"ALL", "Process.EnrichmentAMAS":"false",
"Input.ForceCountryISO3":"AUS", "Input.FormatDelimiter":"CRLF",
"Result.FormatDelimiter":"CRLF", "Input.DefaultCountryISO3":"AUS",
"Result.IncludeInputs":"false", "Result.CountryType":"NAME_EN",
"Process.OptimizationLevel":"STANDARD",
```

```
"Result.PreferredLanguage":"DATABASE", "Process.Mode":"BATCH",
"Result.PreferredScript":"DATABASE", "Result.MaximumResults":"1",
"Result.Casing":"NATIVE", "Database.AddressGlobal":"Database"}};

-- set general configuration
set pb.bdq.uam.global.general.configuration={"cacheSize":"LARGE",
"maxThreadCount":8, "maxAddressObjectCount":8, "rangesToExpand":"NONE",
 "flexibleRangeExpansion":"ON", "enableTransactionLogging":false,
"maxMemoryUsageMB":1024};

-- set unlock codec
set pb.bdq.uam.global.unlockCode=<Insert your Unlock Code here>;

-- set header
set
pb.bdq.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(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(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        |
```

```
+----------+-------------+-----------+---------------------------+--------------+
```

Validate Address Loqate

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.LoqateAddressingUDAF';

-- 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 pb.bdq.uam.loqate.process.configuration={"processType":"VALIDATE",
 "includeMatchedAddressElements":true,
"standardizedInputAddressElements":true, "returnAddressDataBlocks":true,
 "casing":"Mixed", "outputReportSummary":false,
"returnMultipleAddresses":false, "failedOnMultiMatchFound":false,
"countryFormat":"ENGLISH", "defaultCountry":"USA",
"scriptAlphabet":"Native", "returnGeocodedAddressFields":true,
"acceptanceLevel":"Level0", "minimumMatchScore":0,
"formatDataUsingAMASConventions":false,
"singleFieldDuplicateHandling":false,
"multiFieldDuplicateHandling":false,
"nonStandardFieldDuplicateHandling":false,
"outputFieldDuplicateHandling":false, "includeStandardAddress":true,
"duplicateHandling":false, "returnMultipleAddressCount":10};

-- set general configuration
set pb.bdq.uam.loqate.general.configuration={"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 pb.bdq.uam.loqate.engine.configuration={"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 pb.bdq.referencedata.dir=/media/New
Volume/hduser/resources/uam/loqate/Linux;

-- set process type
set pb.bdq.uam.loqate.process.type=VALIDATE;

-- set input header
set pb.bdq.header='InputKeyValue,AddressLine1,AddressLine2,AddressLine3,
AddressLine4,City,StateProvince,PostalCode,Country,FirmName';

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(recordid, 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(recordid, 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

Open Name Parser

Sample Hive Script

```
-- 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 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 at the same path on each node of the cluster.
set refdir='/home/hadoop/reference/';

-- set header
set header='inputrecordid,Name,nametype';

-- 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(${hiveconf:rule},
 ${hiveconf:refdir}, ${hiveconf: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(${hiveconf:rule},
 ${hiveconf:refdir}, ${hiveconf: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           |
+----------------------------+-----+--------------------------------+
```

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 an HDFS directory.
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 `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

C - ISO Country Codes and Module Support

In this section

ISO Country Codes and Module Support

The table lists the two-digit and three-digit ISO codes for each country.

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Afghanistan	AF	AFG
Aland Islands	AX	ALA
Albania	AL	ALB
Algeria	DZ	DZA
American Samoa	AS	ASM
Andorra	AD	AND
Angola	AO	AGO
Anguilla	AI	AIA
Antarctica	AQ	ATA
Antigua And Barbuda	AG	ATG
Argentina	AR	ARG
Armenia	AM	ARM

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Aruba	AW	ABW
Australia	AU	AUS
Austria	AT	AUT
Azerbaijan	AZ	AZE
Bahamas	BS	BHS
Bahrain	BH	BHR
Bangladesh	BD	BGD
Barbados	BB	BRB
Belarus	BY	BLR
Belgium	BE	BEL
Belize	BZ	BLZ
Benin	BJ	BEN
Bermuda	BM	BMU
Bhutan	BT	BTN

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Bolivia, Plurinational State Of	BO	BOL
Bonaire, Saint Eustatius And Saba	BQ	BES
Bosnia And Herzegovina	BA	BIH
Botswana	BW	BWA
Bouvet Island	BV	BVT
Brazil	BR	BRA
British Indian Ocean Territory	IO	IOT
Brunei Darussalam	BN	BRN
Bulgaria	BG	BGR
Burkina Faso	BF	BFA
Burundi	BI	BDI
Cambodia	KH	KHM
Cameroon	CM	CMR
Canada	CA	CAN

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Cape Verde	CV	CPV
Cayman Islands	KY	CYM
Central African Republic	CF	CAF
Chad	TD	TCD
Chile	CL	CHL
China	CN	CHN
Christmas Island	CX	CXR
Cocos (Keeling) Islands	CC	CCK
Colombia	CO	COL
Comoros	KM	COM
Congo	CG	COG
Congo, The Democratic Republic Of The	CD	COD
Cook Islands	CK	COK
Costa Rica	CR	CRI

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Côte d'Ivoire	CI	CIV
Croatia	HR	HRV
Cuba	CU	CUB
Curacao	CW	CUW
Cyprus	CY	CYP
Czech Republic	CZ	CZE
Denmark	DK	DNK
Djibouti	DJ	DJI
Dominica	DM	DMA
Dominican Republic	DO	DOM
Ecuador	EC	ECU
Egypt	EG	EGY
El Salvador	SV	SLV
Equatorial Guinea	GQ	GNQ

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Eritrea	ER	ERI
Estonia	EE	EST
Ethiopia	ET	ETH
Falkland Islands (Malvinas)	FK	FLK
Faroe Islands	FO	FRO
Fiji	FJ	FJI
Finland	FI	FIN
France	FR	FRA
French Guiana	GF	GUF
French Polynesia	PF	PYF
French Southern Territories	TF	ATF
Gabon	GA	GAB
Gambia	GM	GMB
Georgia	GE	GEO

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Germany	DE	DEU
Ghana	GH	GHA
Gibraltar	GI	GIB
Greece	GR	GRC
Greenland	GL	GRL
Grenada	GD	GRD
Guadeloupe	GP	GLP
Guam	GU	GUM
Guatemala	GT	GTM
Guernsey	GG	GGY
Guinea	GN	GIN
Guinea-Bissau	GW	GNB
Guyana	GY	GUY
Haiti	HT	HTI

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Heard Island and McDonald Islands	HM	HMD
Holy See (Vatican City State)	VA	VAT
Honduras	HN	HND
Hong Kong	HK	HKG
Hungary	HU	HUN
Iceland	IS	ISL
India	IN	IND
Indonesia	ID	IDN
Iran, Islamic Republic Of	IR	IRN
Iraq	IQ	IRQ
Ireland	IE	IRL
Isle Of Man	IM	IMN
Israel	IL	ISR
Italy	IT	ITA

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Jamaica	JM	JAM
Japan	JP	JPN
Jersey	JE	JEY
Jordan	JO	JOR
Kazakhstan	KZ	KAZ
Kenya	KE	KEN
Kiribati	KI	KIR
Korea, Democratic People's Republic Of	KP	PRK
Korea, Republic Of	KR	KOR
Kosovo	KS	KOS
Kuwait	KW	KWT
Kyrgyzstan	KG	KGZ
Lao People's Democratic Republic	LA	LAO
Latvia	LV	LVA

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Lebanon	LB	LBN
Lesotho	LS	LSO
Liberia	LR	LBR
Libyan Arab Jamahiriya	LY	LBY
Liechtenstein	LI	LIE
Lithuania	LT	LTU
Luxembourg	LU	LUX
Macao	MO	MAC
Macedonia, Former Yugoslav Republic Of	MK	MKD
Madagascar	MG	MDG
Malawi	MW	MWI
Malaysia	MY	MYS
Maldives	MV	MDV
Mali	ML	MLI

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Malta	MT	MLT
Marshall Islands	MH	MHL
Martinique	MQ	MTQ
Mauritania	MR	MRT
Mauritius	MU	MUS
Mayotte	YT	MYT
Mexico	MX	MEX
Micronesia, Federated States Of	FM	FSM
Moldova, Republic Of	MD	MDA
Monaco	MC	MCO
Mongolia	MN	MNG
Montenegro	ME	MNE
Montserrat	MS	MSR
Morocco	MA	MAR

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Mozambique	MZ	MOZ
Myanmar	MM	MMR
Namibia	NA	NAM
Nauru	NR	NRU
Nepal	NP	NPL
Netherlands	NL	NLD
New Caledonia	NC	NCL
New Zealand	NZ	NZL
Nicaragua	NI	NIC
Niger	NE	NER
Nigeria	NG	NGA
Niue	NU	NIU
Norfolk Island	NF	NFK
Northern Mariana Islands	MP	MNP

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Norway	NO	NOR
Oman	OM	OMN
Pakistan	PK	PAK
Palau	PW	PLW
Palestinian Territory, Occupied	PS	PSE
Panama	PA	PAN
Papua New Guinea	PG	PNG
Paraguay	PY	PRY
Peru	PE	PER
Philippines	PH	PHL
Pitcairn	PN	PCN
Poland	PL	POL
Portugal	PT	PRT
Puerto Rico	PR	PRI

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Qatar	QA	QAT
Reunion	RE	REU
Romania	RO	ROU
Russian Federation	RU	RUS
Rwanda	RW	RWA
Saint Barthelemy	BL	BLM
Saint Helena, Ascension & Tristan Da Cunha	SH	SHE
Saint Kitts and Nevis	KN	KNA
Saint Lucia	LC	LCA
Saint Martin (French Part)	MF	MAF
Saint Pierre and Miquelon	PM	SPM
Saint Vincent and the Grenadines	VC	VCT
Samoa	WS	WSM
San Marino	SM	SMR

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Sao Tome and Principe	ST	STP
Saudi Arabia	SA	SAU
Senegal	SN	SEN
Serbia	RS	SRB
Seychelles	SC	SYC
Sierra Leone	SL	SLE
Singapore	SG	SGP
Sint Maarten (Dutch Part)	SX	SXM
Slovakia	SK	SVK
Slovenia	SI	SVN
Solomon Islands	SB	SLB
Somalia	SO	SOM
South Africa	ZA	ZAF
South Georgia And The South Sandwich Islands	GS	SGS

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
South Sudan	SS	SSD
Spain	ES	ESP
Sri Lanka	LK	LKA
Sudan	SD	SDN
Suriname	SR	SUR
Svalbard And Jan Mayen	SJ	SJM
Swaziland	SZ	SWZ
Sweden	SE	SWE
Switzerland	CH	CHE
Syrian Arab Republic	SY	SYR
Taiwan, Province of China	TW	TWN
Tajikistan	TJ	TJK
Tanzania, United Republic Of	TZ	TZA
Thailand	TH	THA

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Timor-Leste	TL	TLS
Togo	TG	TGO
Tokelau	TK	TKL
Tonga	TO	TON
Trinidad and Tobago	TT	TTO
Tunisia	TN	TUN
Turkey	TR	TUR
Turkmenistan	TM	TKM
Turks And Caicos Islands	TC	TCA
Tuvalu	TV	TUV
Uganda	UG	UGA
Ukraine	UA	UKR
United Arab Emirates	AE	ARE
United Kingdom	GB	GBR

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
United States	US	USA
United States Minor Outlying Islands	UM	UMI
Uruguay	UY	URY
Uzbekistan	UZ	UZB
Vanuatu	VU	VUT
Venezuela, Bolivarian Republic Of	VE	VEN
Viet Nam	VN	VNM
Virgin Islands, British	VG	VGB
Virgin Islands, U.S.	VI	VIR
Wallis and Futuna	WF	WLF
Western Sahara	EH	ESH
Yemen	YE	YEM
Zambia	ZM	ZMB
Zimbabwe	ZW	ZWE

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